

Dr. Dobb's Journal

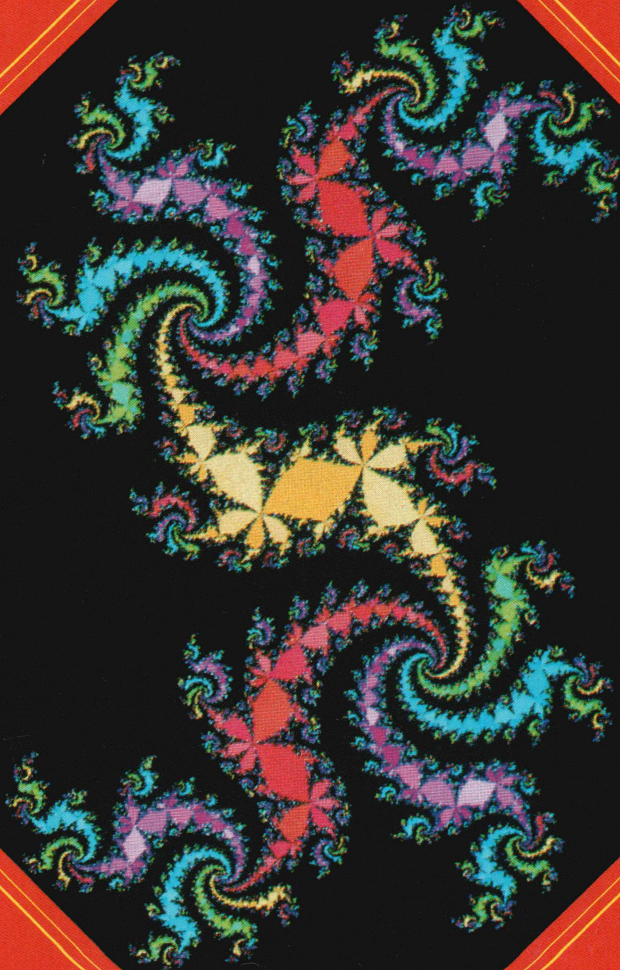
SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS

#103 May 1985 \$2.95 (3.50 Canada)

Graphics Algorithms

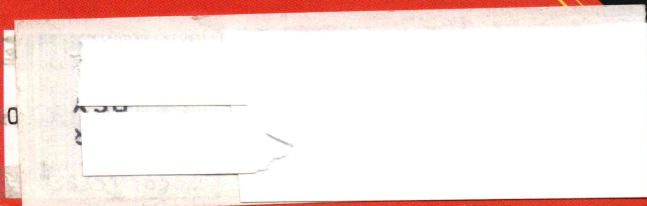
**A C Command-line
Processor**

**PC/AT & Mac Notes
& 68K Math**

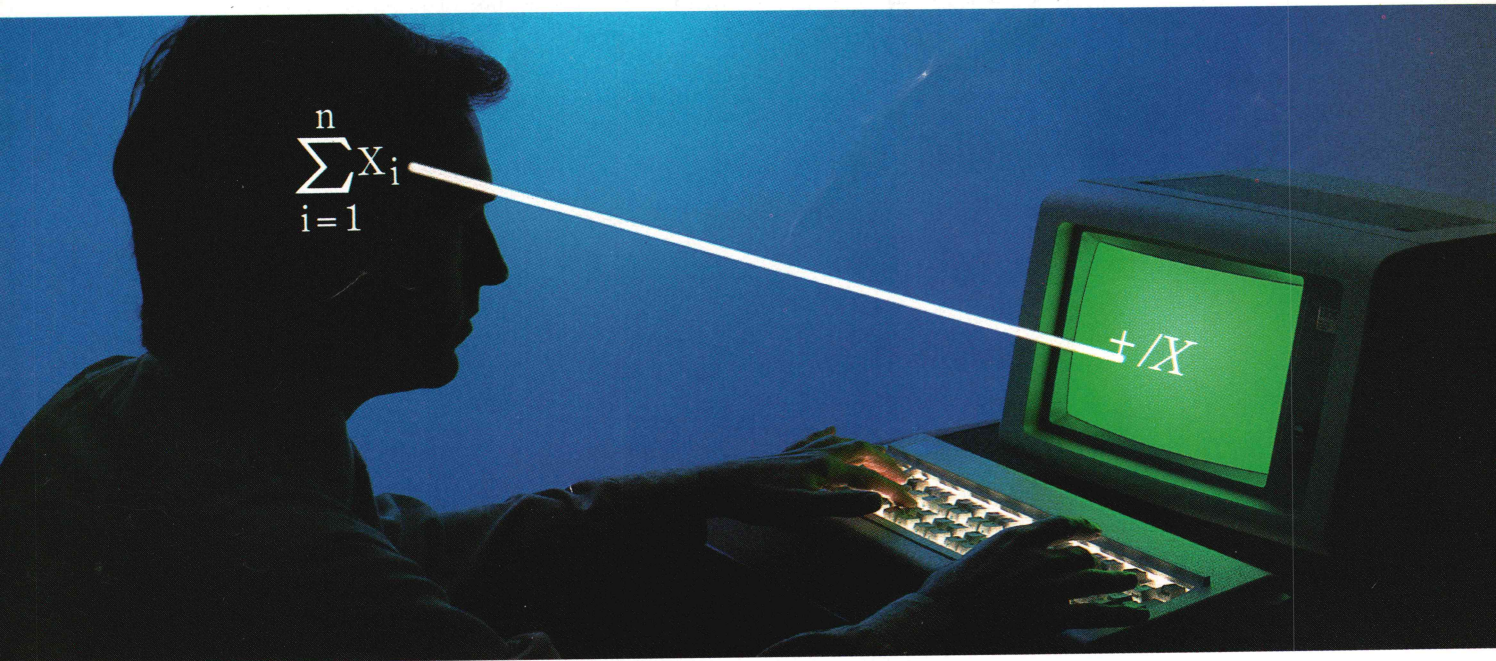


Using Prolog

**Testing
C/80 & Better Basic**



SOLVE PROGRAMMING PROBLEMS THE WAY YOU THINK. PURE AND SYMBOL.



APL★PLUS®/PC IS THE ANSWER.

The shortest distance between two points is a straight line. But unfortunately, that's not the case in programming.

Most languages require you to go through an enormous number of steps before an idea becomes reality.

That's why the APL★PLUS/PC System is such a dramatic and exciting software tool for serious PC programmers and application developers.

Instead of requiring you to learn—and write—long-winded and complicated programs, APL is based on your instinctive ability to deal in symbols. And once you begin using APL's quick notations, you'll find it the ideal programming

environment for all your application needs.

The incredible shortcuts you'll get with APL not only make you more productive, but make programming enjoyable. Intricate calculations and modeling on PC's are a snap. You'll spend less time on drudgery, and more time creating.

Only with APL★PLUS/PC, do you get:

- full-screen editing
- a built-in terminal emulator
- communications
- graphics primitives
- and report formatting.

Writing time-consuming programs like sorting, matrix inversions, and string searching is eliminated. APL's concise notation

already provides these...and more.

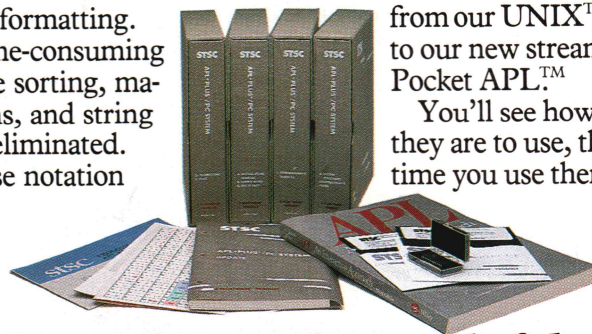
No wonder a *PC Magazine* reviewer enthusiastically reacted to our APL★PLUS/PC System with "awe and delight."

So will you. The complete package price is \$595 and major credit cards are accepted.

Act now and we'll send you a free Convincer Kit. Contact your local dealer, or call **800-592-0050** (in Maryland, call **301-984-5123**) to order your system, or for more information about our other APL PLUS★WARE™ products—

from our UNIX™ version to our new streamlined Pocket APL™.

You'll see how symbol they are to use, the very first time you use them.



Problem-solving at the speed of thought.


STSC
A Contel Company

APL★PLUS/PC System requires 192K. A soft character set can be used for computers with IBM compatible graphics board. A character generator ROM or software is included for the IBM PC or selected compatibles.
PLUS★WARE and POCKET APL are trademarks of STSC, Inc. APL★PLUS is a registered service mark and trademark of STSC, Inc. UNIX is a trademark of AT&T Bell Laboratories.

Circle no. 110 on reader service card.

Speed, Power, Price.

Borland's Turbo Pascal Family.



The industry standard. With more than 250,000 users worldwide Turbo Pascal is the industry's de facto standard. Turbo Pascal is praised by more engineers, hobbyists, students and professional programmers than any other development environment in the history of microcomputing. And yet, Turbo Pascal is simple and fun to use!

Jeff Duntemann, PC Magazine: "Language deal of the century. . . Turbo Pascal: It introduces a new programming environment and runs like magic."

Dave Garland, Popular Computing: "Most Pascal compilers barely fit on a disk, but Turbo Pascal packs an editor, compiler, linker, and run-time library into just 29K bytes of random-access memory."

Jerry Pournelle, BYTE: "What I think the computer industry is headed for: well documented, standard, plenty of good features, and a reasonable price."

Portability. Turbo Pascal is available today for most computers running PC DOS, MS DOS, CP/M 80 or CP/M 86. A XENIX version of Turbo Pascal will soon be announced, and before the end of the year, Turbo Pascal will be running on most 68000 based microcomputers.

\$69.95

High resolution monochrome graphics for the IBM PC and the Zenith 100 computers


Dazzling graphics and painless windows. The Turbo Graphix Toolbox will give even a beginning programmer the expert's edge. It's a complete library of Pascal procedures that include:

- Full graphics window management.
- Tools that will allow you to draw and hatch pie charts, bar charts, circles, rectangles and a full range of geometric shapes.
- Procedures that will save and restore graphic images to and from disk.
- Functions that will allow you to precisely plot curves.
- Tools that will allow you to create animation or solve those difficult curve fitting problems. and much, much more

No sweat and no royalties. You may incorporate part, or all of these tools in your programs, and yet, we won't charge you any royalties. Best of all, these functions and procedures come complete with commented source code on disk ready to compile!

\$54.95

NEW



Searching and sorting made simple


The perfect complement to Turbo Pascal. It contains: **Turbo-Access**, a powerful implementation of the state-of-the-art B+ tree ISAM technique; **Turbo-Sort**, a super efficient implementation of the fastest data sorting algorithm, "Quicksort on disk". And much more.

Jerry Pournelle, BYTE: "The tools include a B+ tree search and a sorting system; I've seen stuff like this, but not as well thought out, sell for hundreds of dollars."

Get started right away: free database! Included on every Toolbox disk is the source code to a working data base which demonstrates how powerful and easy to use the Turbo-Access system really is. Modify it to suit your individual needs or just compile it and run.

Remember, no royalties!

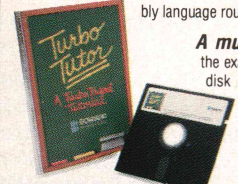
\$54.95



From Start to Finish in 300 pages. Turbo Tutor is for everyone, from novice to expert. Even if you've never programmed before, Turbo Tutor will get you started right away. If you already have some experience with Pascal or another programming language, Turbo Tutor will take you step by step through topics like data structures and pointers. If you're an expert, you'll love the sections detailing subjects such as "how to use assembly language routines with your Turbo Pascal programs."

A must. You'll find the source code for all the examples in the book on the accompanying disk ready to compile. Turbo Tutor might be the only reference on Pascal and programming you'll ever need.

\$34.95



TURBO PASCAL FAMILY

Available at better dealers nationwide. Call (800) 556-2283 for the dealer nearest you. To order by Credit Card call (800) 255-8008, CA (800) 742-1133

Carefully Describe your Computer System!

Mine is: ☐ 8 bit ☐ 16 bit
 I Use: ☐ PC-DOS ☐ MS-DOS
☐ CP/M 80 ☐ CP/M 86
 My computers' name/model is: _____

The disk size I use is:
☐ 3 1/2" ☐ 5 1/4" ☐ 8"

Name: _____
 Shipping Address: _____
 City: _____ Zip: _____
 State: _____ Telephone: _____

Amount: (CA 6% tax) _____
 Payment: VISA MC BankDraft Check
 Credit Card Expir. Date: _____
 Card #: _____

Pascal 3.0 \$ 69.95
 Pascal w/8087 \$109.90
 Pascal w/BCD \$109.90
 Pascal w/8087 & BCD \$124.95
 Turbo Toolbox \$ 54.95
 Turbo Graphix \$ 54.95
 Turbo Tutor \$ 34.95
 *These prices include shipping to all U.S. cities. All foreign orders add \$10 per product ordered.

COD's and Purchase Orders WILL NOT be accepted by Borland. California residents: add 6% sales tax. Outside USA: add \$10 and make payment by bank draft, payable in US dollars drawn on a US bank.

F11

BORLAND
INTERNATIONAL

Software's Newest Direction
4585 Scotts Valley Drive
Scotts Valley, CA 95066
TELEX 172373

Turbo Pascal is a registered trademark of Borland International, Inc.

Circle no. 11 on reader service card.

Up Your ATTM for

\$56 per Megabyte!

■ While our specifications certainly speak for themselves, we thought you still might like to hear from some of our users:

■ "Emerald Systems expands the potential of PCs by providing the ability to access large amounts of data on line, quickly and reliably."
Terry Baptiste, Computerland, Lafayette, Ca.

■ "Service and support is great, which is an unusual experience. Emerald's software for backup and restore is invaluable. Can't put a price on it. Productivity and efficiency has increased at least 50%!"
Bruce Kittinger, Pinon Systems, Ft. Collins, Co.

■ "Runs like a champ with 3-Com Ethernet."
Alvaro Ramirez, Micro Age, Miami, Fl.

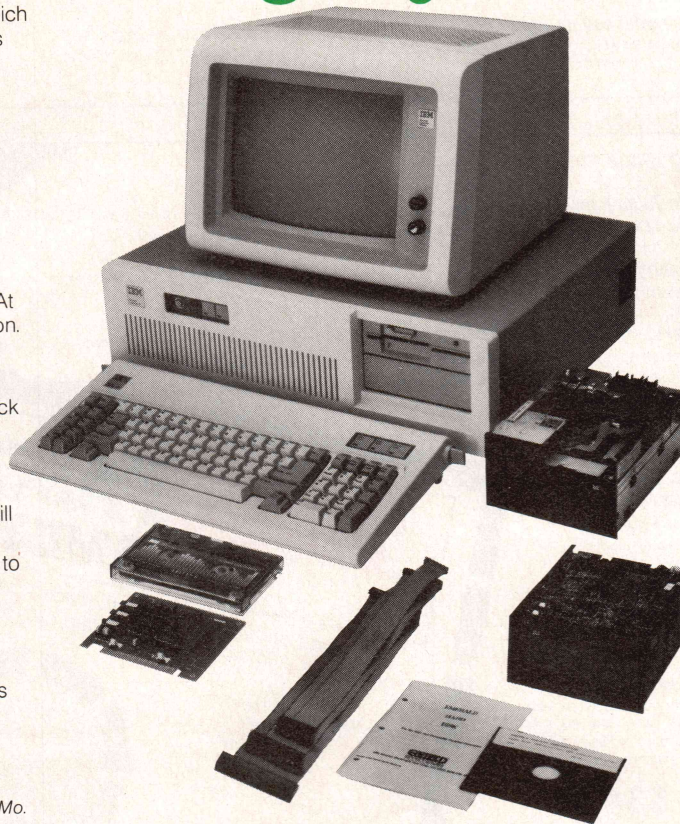
■ "... high capacity and flexibility. At last, a tape back up we can count on. And the price is right!"
John Acres, EDT, Las Vegas, Nv.

■ "The speed at which you can back up is very impressive."
Jim McEwen, Mercy Hospital, Portland, Me.

■ "When Emerald says your unit will be there on Thursday, it's there on Thursday! Delighted we were able to exceed the usual 32 megabyte restriction."
Steven Mayer, Take One Company, New York, N. Y.

■ "The Emerald 70 MB hard disk is extremely easy to install and work with. Emerald has a complicated piece of equipment made easy to use."
Tom Edler, Jewish Hospital, St. Louis, Mo.

■ "Our Emerald fixed disk installed quickly and easily. Emerald's reliable disk and tape backup further enhances LIBRA's high function accounting software."
Kenn White, Libra Programming, Salt Lake City, Ut.



HARD DISK

We've broken through the 32 MByte DOS barrier!

Now you can create up to 240 MByte databases on one file, with multiple volumes per disk drive. You get 14 times more storage with a 30% increase in access speed!

And that's not all:

Expands up to 280 MB for as little as \$56 per megabyte.
6 expansion slots
Up to 24 volumes
Drive sizes: 40,70,140
Internal or External
Also for PC,XT, and compatibles
Supports all PC compatible networks
Simple menu-driven installation

TAPE BACKUP

1/4" cartridge
1/2" 9 track
60 Megabytes
No lost data from tape run out
Backup and Restore Utility (BRU)TM
software included
LAN Compatible*

***FREE APPLICATION GUIDE:**
"IBM LAN Installation and Implementation."

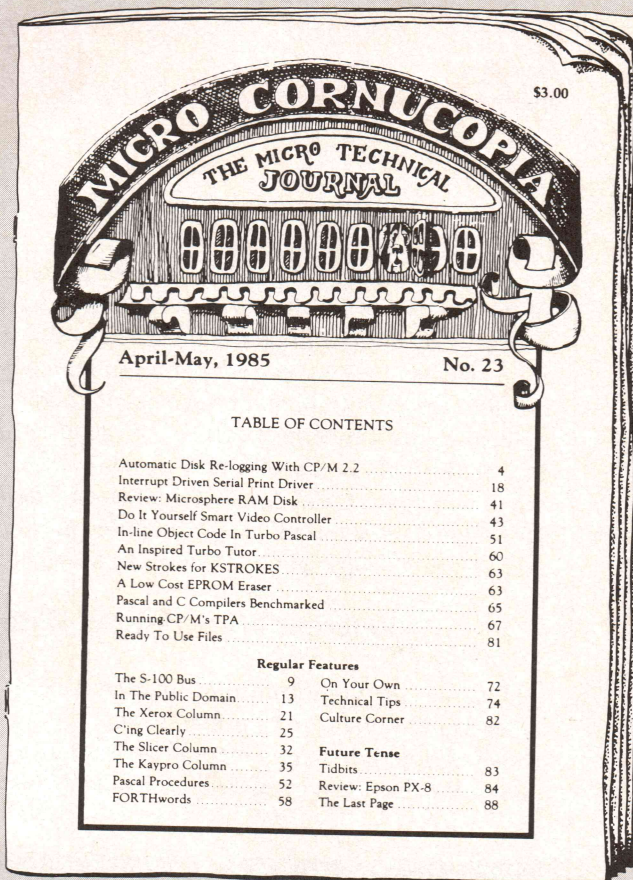
**Call (619) 270-1994,
or write to
EmeraldTM Systems
Corporation**

**Mainframe Storage for Micros
4901 Morena Blvd,
San Diego, 92117
TLX 323458 EMERSYS
EASYLINK 62853804**

Distributed by Manchester Equipment of NYC, and selected Entre, MicroAge and Computerland stores.

Emerald, BRU, Up Your AT, and Mainframe Storage for Micros are registered trademarks of Emerald Systems Corporation. IBM PC/XT/AT are registered trademarks of IBM Corporation.

EMERALDTM
SYSTEMS CORPORATION
Mainframe Storage for MicrosTM



RISK FREE

You get your 7th issue free when you order a year subscription (6 issues) for \$16. Plus, if you're not delighted with Micro C after receiving your first issue, just drop a note and we'll refund your entire \$16, no questions asked.

Order # (503) 382-5060
! SPECIAL OFFER !

Quantity	Description	Price	Total
	Year Subscription (plus 7th issue free)	\$16.00	
	Canada & Mexico	\$22.00	
	Other Foreign	\$30.00	
(US Funds only, payable on a US Bank)			

NAME _____

ADDRESS _____

CITY _____

STATE _____

ZIP _____

Only **\$16.00**

Make check or money order payable to:

Micro Cornucopia

Magazine

P.O. Box 223

Bend, OR 97709

☐ VISA

☐ MasterCard

Card No. _____

Exp. _____

Signature _____

Dr. Dobb's Journal

Editorial

Editor-in-Chief *Michael Swaine*
Editor *Randy Sutherland*
Managing Editor *Frank DeRose*
Technical Editor *Alex Ragen*
Editorial Assistant *Sara Noah*
Contributing Editors *Robert Blum,*
Dave Cortesi,
Ray Duncan,
Allen Holub
Copy Editor *Rhoda Simmons*
Typesetter *Jean Aring*

Production

Design/Production
Director *Detta Penna*
Art Director *Shelley Rae Doeden*
Production Assistant *Alida Hinton*
Cover *Benoit Mandelbrot*

Advertising

Advertising Director *Stephen Friedman*
Advertising Sales *Walter Andrzejewski,*
Shawn Horst,
Beth Dudas,
Michele Beaty
DDJ Classifieds *Alice Abrams*
Advertising Coordinators *Lisa Boudreau*
Jay Horvath

Circulation

Fulfillment Mgr. *Stephanie Barber*
Subscription Mgr. *Maureen Snee*
Book Marketing Mgr. *Jane Sharninghouse*
Single Copy Sales Mgr. *Kathleen Boyd*

Administration

Finance Manager *Sandra Dunie*
Business Manager *Betty Trickett*
Accounts Payable Supv. *Mayda Lopez-Quintana*
Accounts Payable Asst. *Denise Giannini*
Billing Coordinator *Laura Di Lazzaro*

M&T Publishing, Inc.

Chairman of the Board *Otmar Weber*
Director *C.F. von Quadt*
President *Laird Foshay*

Entire contents copyright © 1985 by M&T Publishing, Inc. unless otherwise noted on specific articles. All rights reserved.

Dr. Dobb's Journal (USPS 307690) is published monthly by M&T Publishing, Inc., 2464 Embarcadero Way, Palo Alto, CA 94303, (415) 424-0600. Second class postage paid at Palo Alto and at additional entry points.

Address correction requested. Postmaster: Send Form 3579 to *Dr. Dobb's Journal*, 2464 Embarcadero Way, Palo Alto, CA 94303. **ISSN 0278-6508**

Subscription Rates: \$25 per year within the United States, \$44 for first class to Canada and Mexico, \$62 for airmail to other countries. Payment must be in U.S. Dollars, drawn on a U.S. Bank. For subscription problems, call: outside of CA 1-800-321-3333; within CA 1-619-485-6535.

Foreign Distributors: ASCII Publishing, Inc. (Japan), Computer Services (Australia), Computer Store (New Zealand), Computercollectief (Nederland), Homecomputer Vertriebs GMBH (West Germany), International Presse (West Germany), La Nacelle Bookstore (France), McGill's News Agency PTY LTD (Australia), Progreso (France).

People's Computer Company

Dr. Dobb's Journal is published by M&T Publishing, Inc. under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a non-profit, educational corporation.

May 1985
Volume 10, Issue 5

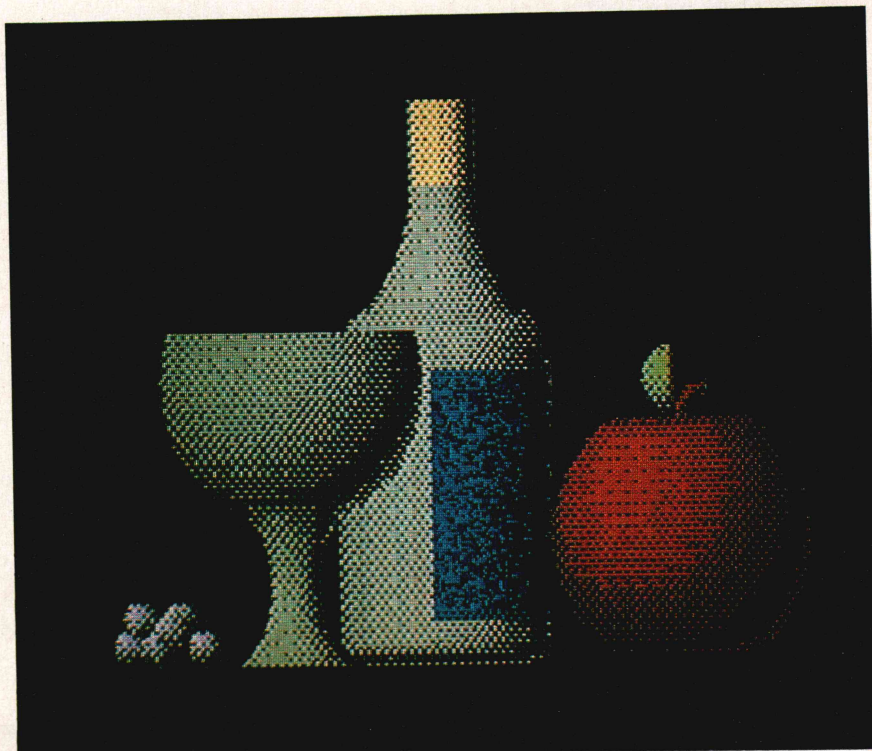
CONTENTS

Cover Artist

Our cover "Fractal Dragon," is the creation of Benoit Mandelbrot, Professor of the Practice of Mathematics at Harvard University. The most conspicuous characteristic of this dragon is that the eye and the imagination distinguish in it, in addition to the overall shape, an infinity of smaller features of every size down to the infinitesimal. For geometric shapes of this kind, Professor Mandelbrot coined the term "fractal." "Fractal Dragon" is a portion of "attractor" that occurs in a problem of dynamics. The cover illustration is a reproduction of Plate C5 of Mandelbrot's book *The Fractal Geometry of Nature* (Freeman 1982), drawn using computer programs by Mark R. Laff and V. Alan Norton. Copyright 1982 by B. B. Mandelbrot.

Referees Who Assisted With This Issue

Ted Carnevale, State University of New York at Stony Brook
Allen Holub, *DDJ* Contributing Editor
Robert Tripp, Editor-in-Chief of the late *MICRO*
James Woormer, Commodore-64 Guru and Gazebo Builder



See Richard Rylander's "Solid Shape Drawing on the Commodore 64" page 50.

Dr. Dobb's Journal

ARTICLES

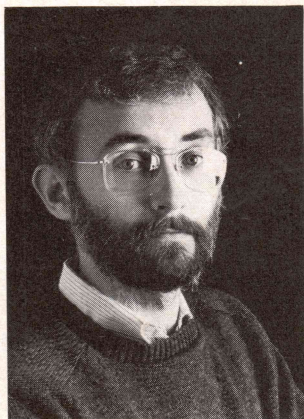
- | | | |
|--|--|---|
| <p>Using Decision Variables in Graphics Primitives
by Tom Hogan</p> <p>Solid Shape Drawing on the Commodore 64
by Richard Rylander</p> <p>A Compiler Written in Prolog
by G. A. Edgar</p> | <p>40</p> <p>50</p> <p>84</p> | <p>An algorithm that uses the Decision Variable Method for plotting ellipses. (Reader Ballot No. 193).</p> <p>Some new approaches to computer-generated images on small systems. Combining elementary shapes, using the polygon mesh technique, creating shading effects. (Reader Ballot No. 194).</p> <p>A compiler for VALGOL I written in Micro-PROLOG. (Reader Ballot No. 195).</p> |
|--|--|---|

COLUMNS

- | | | |
|---|--|--|
| <p>Dr. Dobb's Clinic
by D. E. Cortesi</p> <p>Realizable Fantasies
by D. E. Cortesi</p> <p>C Chest
by Allen Holub</p> <p>16-Bit Software Toolbox
by Ray Duncan</p> | <p>16</p> <p>22</p> <p>28</p> <p>118</p> | <p>A wood blocks program for the Atari; keyboards with N-key rollover; DOS services and device drivers. (Reader Ballot No. 190).</p> <p>The Resident Intern reflects on hackers, radicals, the microcomputer revolution and the role that DDJ will play in fostering that revolution. "Down with Cybercrud!" (Reader Ballot No. 191).</p> <p>A general purpose command line processor. (Reader Ballot No. 192).</p> <p>68000 square root routine; MacFeedback; MSDOS device drivers; Microsoft Assembler Bug of the Month. (Reader Ballot No. 196)</p> |
|---|--|--|

DEPARTMENTS

- | | | |
|--|---|---|
| <p>Editorial</p> <p>Letters</p> <p>Reviews</p> <p>DDJ Classifieds</p> <p>Advertiser Index</p> | <p>6</p> <p>8</p> <p>98</p> <p>97</p> <p>128</p> | <p>Software Toolworks' C/80 and C/80 Mathpack; AMPRO's Little Board and Bookshelf Computer; New Generation Systems' Diskmaker I; Summit Software's BetterBASIC.</p> |
|--|---|---|



I want to comment on what others have been saying about two subjects germane to this or any issue of *DDJ*: graphics and copy-protected disks.

Frank Gaude regards the iconic interface of the Mac and its imitators as a regression to preliterate cave-scratchings. In his *Z-News* he says, "Long ago we used hieroglyphs to communicate. Then after a struggle an alphabet developed. Now we go back to pictures: icons. How many glyphs can we remember?"

In the macaphony of the current iconolatry this cautionary message is worthy of attention, but I suggest that it's not the whole story. The function of software is to realize metaphors. Icons are capable of doing that well but are no help in stepping outside existing metaphors to invent new ones; no help, that is, in developing new software. I would agree that icons are mnemonic for users and superfluous for programmers and let it go at that, but for the fact that I have recently seen a prototype of a system that takes icons a step further, a system in which actions can be assigned user-created visual tokens, and can be combined in ways that visually and kinaesthetically convey the form of the combination. Active, growing icons.

Furthermore, while the Mac icons are static tokens rather than elements of a true language, that is not true of all computer graphic aids to communication. French researchers are currently developing the means for transmitting sign language over telephone lines at commercial data rates. Their idea is to extract from a sequence of video frames just the essential components of a gesture and pass those along. This does not, I should make clear, mean matching the gesture to some template from a database of acceptable gestures; rather it means producing a real-time cartoon version of the gesturing party. The cartoons, subject to idiomatic and dialectical variations, are truly linguistic graphic elements, which is just what icons aren't.

As for copy-protected disks

"... imagine yourself buying a new car. You have narrowed your choice to two cars, identical in all respects except that one will travel 100 mph and the other has a governor on it and cannot go past 55. Which car would you buy?"—Michael D. Brown, Central Point Software, producer of Copy II.

"People can rest assured that...we will always offer versions of our products that are not copy-protected."—Philippe Kahn, Borland International.

"Companies like Multimate and Borland have the right idea. Lotus, Ashton-Tate and MicroPro are the bad guys."—Edward Messerly, SF regional administrator of the GSA.

"... a growing number of publishers are dropping traditional copy-protection methods, which they conclude have created ill-will and inconvenience for users Among the publishers who have abandoned copy-protection schemes is MicroPro Many software publishers, however, hope to see new protection methods"—Time/Life Access:Apple.

"These people are nutty."—Edward Messerly.

Yes. Something is nutty when a vendor can brag about also selling non-broken goods. It's really simple, isn't it? Vendors of software who want to stay vendors of software must find ways to profit from their efforts without breaking the tools they sell. The ability to copy files and disks is a facility I bought with my hardware and operating system. To try to sell me a program designed to subvert my system and hamper its operation is to try to enlist me in vandalism against my own property. Even granting that copy-crippling is a morally defensible policy, how could anyone believe that it was commercially viable in the long run? Nutty.

Michael Swaine

Michael Swaine

Another in a series of
productivity notes on MS-DOS™
software from UniPress.

**Subject: Multi-window full
screen editor.**

Multiple windows allow several files
(or portions of the same file) to be
edited simultaneously. Program-
mable through macros and the built-
in compiled MLISP™ extension
language.

Features:

- Famed Gosling Version.
- Extensible through the built-in
MLISP programming language and
macros.
- Dozens of source code MLISP
functions; including C, Pascal and
MLISP syntax checking.
- EMACS runs on TI-PC™, IBM-PC AT™,
DEC Rainbow™ or any other MS-DOS
machine. Requires at least 384k.
- Run Lattice® C or PsMake™ in
the background and EMACS will
point to any errors for ease of de-
bugging. PsMake is a UNIX™-style
"make" utility to automate the proc-
ess of building complex programs.
- Optional Carousel Tools: UNIX-
like facilities including cat, cp, cd,
logout, ls, mv, pwd, rm, set, sh
and more.

Price:

EMACS	\$475
One month trial	75
Available for UNIX and VMS.	
PsMake	179
Carousel Tools	149
Full System	1,299
Includes EMACS (object), PsMake, Lattice C, PHACT™ ISAM and Carousel Tools.	

UNIPRESS EMACS™

COMPILER
FOR THE 8086™ FAMILY

LATTICE® C COMPILER

Features:

- Runs on the IBM-PC™ under
MS-DOS 1.0, 2.0 or 3.0
- Produces highly optimized code.
- Small, medium, compact and
large address models available.
- Standard C library.
- PLINK—optional full function
linkage editor including overlay
and support.

Price:

Lattice C Compiler	\$425
PLINK	425

Subject: Compiler for MS-DOS.

Lattice C Compiler is regarded as
the finest compiler for MS-DOS and
is running on thousands of 8086
systems.

Features:

- Supports fixed and variable length
records (1-9999 bytes).
- Up to 9 alternate indices are sup-
ported.
- Record locking allows each record
in the database to allow multiple
simultaneous updates.
- Records can be accessed on full
or partial key.
- Includes full Lattice linkable library
and high-level functions.
- Optional, PHACT-rg, a powerful
and flexible report generator which
provides a high level, easy-to-use
command language for formatting
reports from existing PHACT data-
bases. Available for UNIX, MS-DOS
and VMS.

Price:

PHACT ISAM	\$250
PHACT-rg	165
Source Code available, call for terms.	

ISAM FILE SYSTEM
& REPORT WRITER

PHACT™

**Subject: Powerful Keyed File
Access for MS-DOS.**

PHACT ISAM is a keyed B+ tree
file manager providing easy access
to and manipulation of records in
a database.

For more information on these and
other UNIX software products, call or
write: UniPress Software, Inc., 2025
Lincoln Hwy., Edison, NJ 08817.
Telephone: (201) 985-8000. Order
Desk: (800) 222-0550 (Outside NJ).
Telex: 709418. Japanese Distributor:
Softec 0480 (85) 6565. European Dis-
tributor: Modulator SA (031) 59 22 22.

OEM terms available.
Mastercard/Visa accepted.

Trademarks of Lattice, Lattice, Inc.: UniPress EMACS and MLISP. UniPress
Software, Inc.: MS-DOS, Microsoft, UNIX, AT&T Bell Laboratories, Carousel Tools,
Carousel MicroTools, Inc.: PHACT, PHACT Associates, 8086, Intel, TI-PC, Texas
Instruments, IBM-PC/AT, International Business Machines, DEC Rainbow/VMS,
Digital Equipment Corp.

UniPress Software
Your Leading Source for UNIX Software.



Anorexia Cured

Dear DDJ:

I read with some interest the article 'Fatten Your Mac' that appeared in your January [1985, #99] issue and forthwith did a fattening on my Macintosh. There are a number of points that I would like to add to the article, however. Firstly, a fine supplier of the 256K DRAM chips was omitted. I refer to Microprocessors Unlimited, of Beggs, Oklahoma, (918) 267-4961. I was able to phone them late Friday, and the chips they sent out on Monday arrived here (in Seattle) on Tuesday. I was quite pleased with the price and the service. They accept Visa, and they even sent a brief note explaining proper handling of the chips.

Secondly, it is *not* necessary, though it may be desirable, to destroy the 64K DRAM chips. I desoldered the original chips with an Ungar model 2000 desoldering tool, and was able to remove the solder, let the board cool, then pry the chips from the motherboard, all without clipping a single lead. This damaged two printed circuit traces (pulled them off in the prying process); as the article recommends, *there is a safer way to proceed*. My technique cannot be advised in the absence of an excellent desoldering tool, or for those not confident in their abilities to make printed circuit repairs. The article had so fired my enthusiasm that I yanked out the 64K DRAMs days before the 256K DRAM circuits arrived. I was able to verify the repairs, and the removed memory chips, by installing sockets, plugging the 64K DRAMs into those sockets, and using the Macintosh. The nature of the lifted printed circuit traces (they showed copper color on the ICs to which the traces stuck) made detection easy,

and no faults were missed. The board worked on the first try.

Thirdly, the multiplexer can be installed either piggybacked on another chip, or can be placed on a separate board, attached on the seven solder pads referred to in the article. Those seven pads include all logic signals and power required for the multiplexer. I installed wire-wrap pins in those pads, made a small board for the multiplexer from .100-inch hole-grid breadboard material, and pushed the small board over the wire-wrap pins, wire-wrapping the connections atop it to hold it in place. I found this less tricky than the piggyback soldering job would have been, and no leads of the multiplexer needed to be bent.

Fourthly, the multiplexer recommended is 74F253 or 74AS253; while these will do quite well, so will the 74F153 or 74AS153; which are slightly less expensive; the only difference is the nature of the DISABLE function, which (in this application) is always unused. All pin connections are identical to the '253.

I found that the multiple-layer board, having extra copper in the interior layers, was rather more difficult than most to desolder; I had to apply more heat (use a higher temperature on the desoldering iron) than usual, to complete the solder removal within a safe time. I did not notice any fragility of the printed circuit (no more than most, at any rate), and my procedure was, as I have mentioned, rather stressful. At one point, I fumbled, and dropped the board onto the concrete floor—there was no damage.

Prices were better than I expected, about \$220 total cost for the chips, and that figure is dropping almost daily. Time required was almost exactly four hours. Since Apple charges

about six hundred dollars more for the 512K Mac, it hardly pays to buy one; the fine folks in Cupertino are willing, in effect, to pay me handsomely for those hours, if I convert my 128K instead.

Instead of 'Macintosh' and 'Fat Mac', I will be using the terms 'Macintosh' and 'Anorexic Mac' in the future.

Sincerely grateful,
John Whitmore
FM-15 Physics Dept.,
University of Washington
Seattle, WA 98195

Going PUBLIC

Dear DDJ:

PUBLIC files have been used successfully on a wide variety of CP/M computers since the publication of our November 1984 [DDJ, #97] article "CP/M 2.2 goes PUBLIC." However, we want again to caution readers that PUBPATCH must be installed on an *unmodified* CP/M 2.2 BDOS!

Several users have discovered that their Heath Co. CP/M 2.2.03 has been altered with a patch at 06E1 and 0DEE-0DF3 (relative to the start of the BDOS). The Heath patch causes the BDOS to stop building the disk group-allocation vector when it encounters the first directory entry having deleted data (E5) codes in both bytes 0 and 1. If PUBPATCH is then installed in this BDOS, the symptoms will be files that disappear from the end of the disk directory!

Heath users with this problem should patch location 06E2h back to its original value '06D2h' (relative) before installing PUBPATCH.

RELATIVE ADDRESS	06E1'
ORIGINAL 2.2 BDOS	JZ 06D2'
HEATH 2.2.03 BDOS	JZ 0EEE'

Two readers have reported problems with PUBLIC.ASM, used to set or reset the PUBLIC file attribute bit. Updated code can be found in [Listing One (page 13)]. The first fix corrects a bug in v. 1.0 that prevents PUBLIC from setting the attribute for a very large file. The second takes care of a bug in the standard CCP (but not in ZCPR), which fails to restore the default dma address before continuing to execute a SUBMIT file.

No changes are required to PUB-PATCH itself.

Sincerely,
Bridger Mitchell, Derek
McKay
Plu*Perfect Systems
Box 1494
Idyllwild, CA 92349

Turbo Bug

Dear DDJ:

I have been using Borland International's excellent Turbo Pascal compiler for several months now (I started with version 1) and I think it is by far the best implementation of any language I have every seen. The quick compiler and built-in editor have reduced the pain of correcting typos far enough to let me appreciate the virtues of Pascal. In the process I have discovered two things about the language that may be of interest to others. I am including listings of three short demo programs.

First, the "initialized constants" mentioned briefly in both the reference manual and the Turbo Tutor manual are really initialized static variables. This is clear from a close reading of either manual, but it is never stated in so many words, nor do any of the example programs demonstrate it. Compiling and running the [program in Listing Two (page 13)] will demonstrate that the typed constant I in the procedure ShowStatic behaves exactly like a static variable in C.

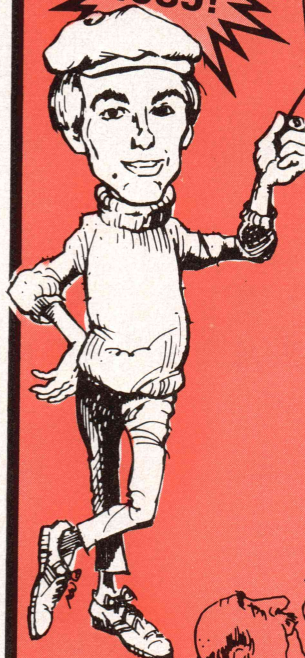
Compiling and running the program [in Listing Three (page 14)] will demonstrate a problem I discovered while trying to process the files produced by a linker cross reference utility. The Trunc and Round functions in both the MSDOS and CP/M versions of Turbo Pascal produce a run time

Are You In XTC™ Yet?

The Ultimate Programmer's Editor

Some folks have already discovered it. And they threw their other editors away! Why? Because XTC is incredibly powerful. It's also easy to learn, and easy to use. XTC has MORE editing facilities for LESS MONEY — 99 bucks

New
for
1985!



JUST LOOK AT THESE LUXURY FEATURES:

WINDOWS — 80 columns wide, independently 4-way scrollable, and non-overlapping. Define 'em the way you want to see them on your screen.

MACROS — Plenty of room for over 100 user-definable macro programs — you can assign 'em to function keys or labels up to 80 characters long!

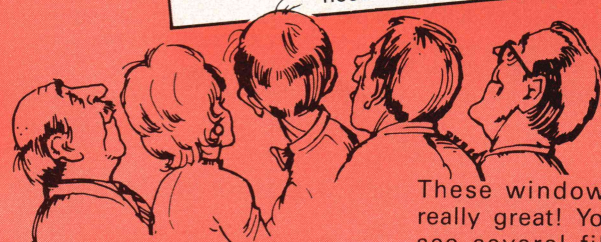
KEYPAD EDITING — Standard where we come from. But for you mavericks, you can redefine those arrows to do auto-indenting, reformatting, the works! Need Wordstar compatibility? You can use your Wordstar editing commands here.

MULTITASKING — All of your macros can run in the foreground or independently in the background as separate processes while you continue editing.

CONTROL STRUCTURES — We've got everything, including IF THEN ELSE, WHILE, REPEAT, FOR, and BREAK.

EDITOR VARIABLES — Your macros can use plenty of variables to do just about anything. You get INTEGERS, BOOLEANS, and STRINGS, plus . . .

TEXT BUFFERS — More than you'll ever need — 20 in fact.



INTRODUCTORY OFFER

Want to compare XTC with your editor? Just ask for our demo disk (only \$5.00) and try it out. When you buy XTC, we'll knock five bucks off the price.

XTC outperforms any other programmable editor on all IBM /PC, /XT, and /AT computers (and true compatibles). XTC even works with your Sidekick and Turbo Pascal from Borland!

To get your copy of XTC now, order it over the phone — we can ship it the same day! Or, you can send in an order, just like this one:

XTC 99 bucks
Shipping and Insurance 3.50
Wash. res. add tax: 7.99
Want it COD? Add this: 1.65
TOTAL IT UP, AND SEND IT QUICK!

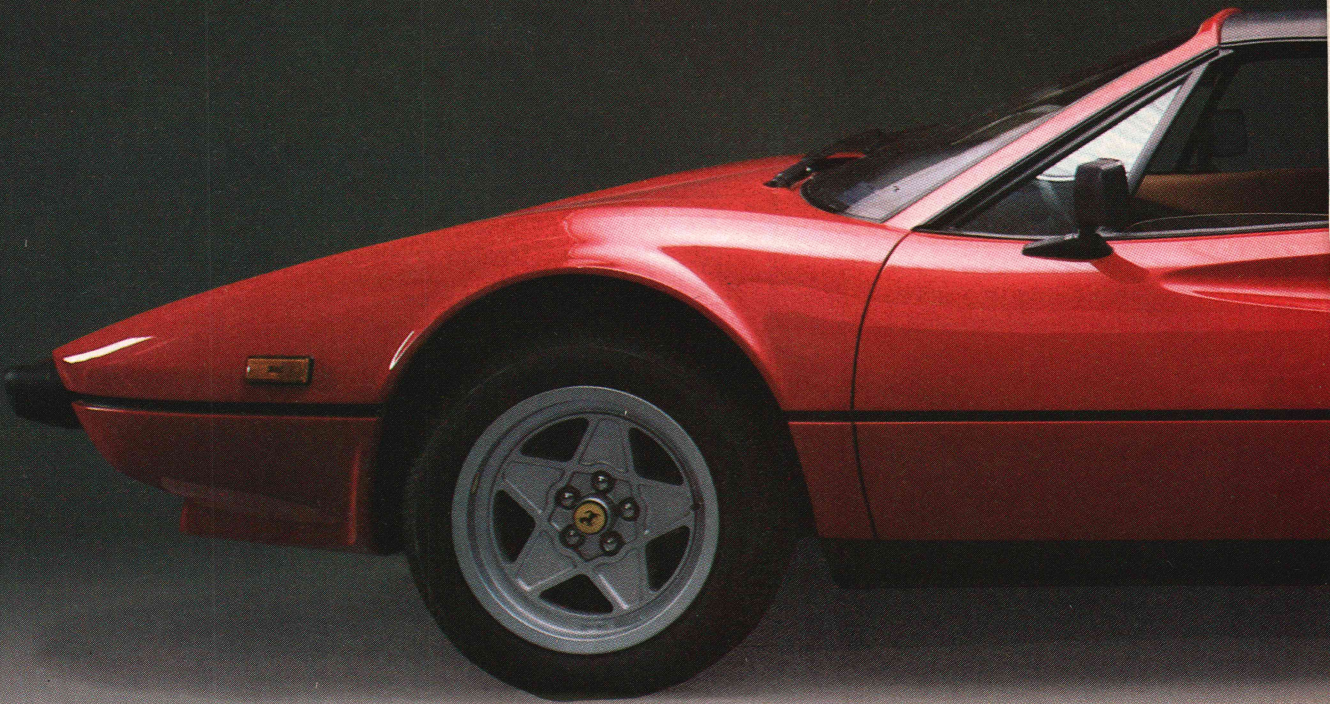
These windows are really great! You can see several files at once — even different parts of the same file. That means you've got declarations in front of you while you're looking at the code that uses them!

Is XTC protected? Heck no! We even give you the source on a disk for your recreation — 7,000 lines of Pascal!

WENDIN™

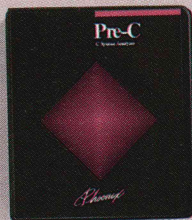
Box 266 • Cheney, WA 99004 • USA • (509) 235-8088

Sidekick and Turbo Pascal are trademarks of Borland, International. Wordstar is a trademark of MicroPro. Wendin and XTC are trademarks of Wendin, Inc.



Programmers' Pfantas

Phoenix makes programmers' dreams come true. With the best-engineered, highest performance programming tools you can find. A full line of MS™-DOS/PC DOS programs and utilities no other company offers. All designed to help you write, test and deliver the best programs possible. Top-of-the-line quality at a price you can afford.



Finally, A Lint For MS-DOS.

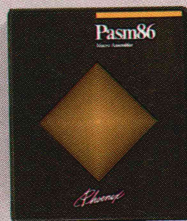
Now you can get the full range of features C programmers working in UNIX™ have come to expect from their Lint program analyzer.

With Pre-C™ you can detect structural errors in C programs five times faster than you can with a debugger. Find usage errors almost impossible to

detect with a compiler. Cross-check multiple source files and parameters passed to functions. Uncover interface bugs that are difficult to isolate. All in a single pass. Capabilities no C compiler, with or without program analyzing utilities, can offer. In fact, Pre-C outlits Lint, since you can handle analyses incrementally.

And, Pre-C's flexible library approach lets you maintain continuity across all the programs in your shop, whether you use Pre-C's pre-built libraries, pre-existing functions you already have, or some you might want to buy yourself.

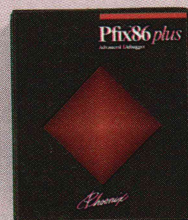
Plus, you're not limited to one particular library, and Pre-C keeps track of all the libraries you're using to make sure that code correctly calls them. **\$395.**



Assemble Programs Twice As Fast.

Pasm™ 86 assembles Masm files 2 to 3 times faster than Masm 3.0. Pasm86 supports 8086/88, 8087, 80186 and 80286 processors.

With Pasm 86's built-in defaults, you can write code quickly since you won't spend hours learning all the control statements needed at the beginning of your program. You can define symbols on the command line. Decide whether you want error messages or not. And, put local symbols within procedures. **\$295.**



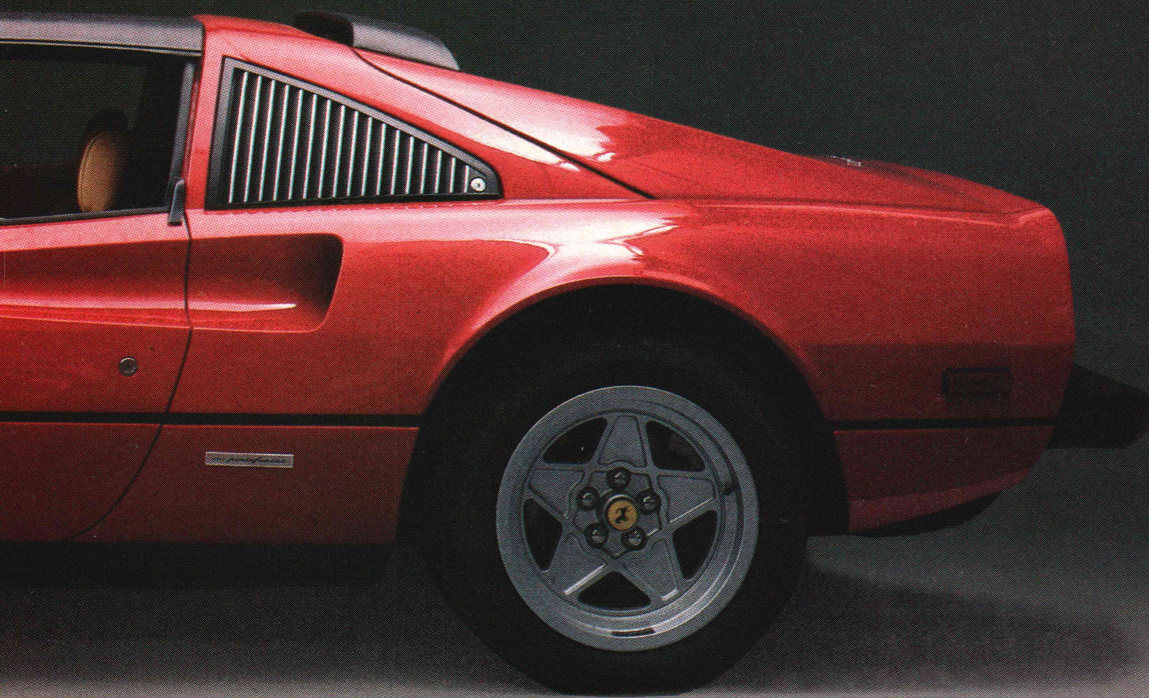
Still Fixing Bugs The Hard Way?

Pfix™ 86 Plus, the most advanced symbolic debugger on the market, eliminates the endless error searches through piles of listings. Locate instructions and data by symbolic name, using symbolic addresses. Handle larger, overlaid programs with ease.

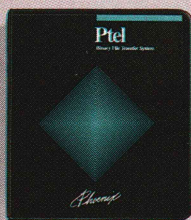
An adjustable multiple-window display shows source and object code and data, breakpoint settings, current machine register and stack contents simultaneously. An in-line assembler allows program corrections directly in assembly language. Powerful breakpoint features run a program full speed until a loop has been performed n times.

With a single keystroke you can trace an instruction and the action will be immediately reflected in source, object, data, stack, and register windows. Another key begins a special trace mode that executes call and loop instructions at full speed. Designed to work with both Plink86 and MS™ LINK linkage editors. **\$395.**

Special Thanks to Gasten Audrey of Framingham, MA.



ies by Phoenix.

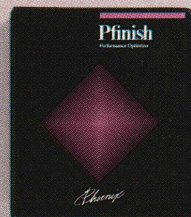


Get The Lead Out Of Binary File Transfer.

Ptel™ is the universal binary file transfer program for MS-DOS 2.0 or 3.0. You can move binary files fast and accurately. Upload or download groups of files from Bulletin Boards or remote computers. Move files between dissimilar machines and operating systems.

Ptel's advanced binary protocol, Telink, offers better-than-Modem7 accuracy and performance. Faster transfer speeds. An on-screen update of error correction, blocks, transferred, and time to complete.

Includes popular Modem7 and XModem protocols. With Checksum or CRC. Plus Kermit and ASCII. **\$195.**

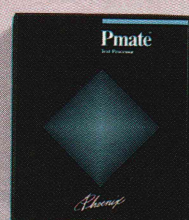


Maximize Your Program's Efficiency.

Pfinish™ delivers the fastest running programs possible. This performance analyzer lets you "zoom in" on the inefficient parts of your program. Whether written in assembly language, C, Pascal, Fortran. Even Basic. Unlike profilers available today, Pfinish under-

stands the structure of your program and reports the amount of activity and time spent in its subroutines or functional groups. Pfinish analyzes both overlaid and memory resident programs. Down to the instruction level. Reports are displayed. Stored on disk. Or printed out. In tabular form or histograms.

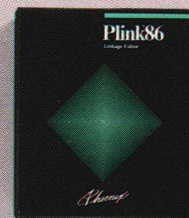
Do a dynamic program scan. Identify the most frequently executed subroutines. Find inefficient code that costs your program valuable time. Rank subroutines by execution frequency. **\$395.**



Why Work With A Primitive Editor?

More than a powerful editor, Pmate™ is a text processing language. An emulator of other editors. A language-specific editor for C, Pascal, and Fortran. Pmate™ can even run in the background!

You get full-screen, single-key editing. Ten editing buffers. Horizontal and vertical scrolling. A "garbage stack" buffer. A built-in macro language with variables, control statements, radix conversion, tracing and 120 commands that you can group and execute with a single keystroke. **\$225.**



Why Squeeze Your Program More Than You Have To?

The Plink™86 overlay linkage editor brings modular programming to 8086/88-based micros. Write large and complex programs without worrying about memory constraints. Work on modules individually, link them into executable files. Use the same module in different programs. Changes the overlay structure of an existing program without recompiling. Use one overlay to access code and data in other overlays.

Plink86 links Intel-format modules. **\$395.**

Call (1) 800-344-7200. In Massachusetts (617) 762-5030. Or, write.

Phoenix Computer Products Corp.
1420 Providence Highway Suite 115
Norwood, MA 02062

Debugging Bugging You?

Torpedo program crashes and debugging delays with debugging dynamite for the IBM PC ...

UP PERISCOPE!

First, you install the hardware.

The hardware's a special memory board that fits in a PC expansion slot. Its 16K of write-protected memory contains Periscope's resident symbolic debugger. No runaway program, however berserk it may be, can touch this memory!

Then you UP PERISCOPE.

Use Periscope's push-button break-out switch to interrupt a running program ... even when the system's hung! Periscope supports Assembly, BASIC, C and Pascal. In addition to the usual debugging capabilities, some of Periscope's features are:

Stop your system in its tracks at any time.

Use symbol names instead of addresses.

Run a program on one monitor and debug on another.

Monitor your program's execution with Periscope's comprehensive breakpoints.

Debug memory-resident programs.

Put your time to better use.

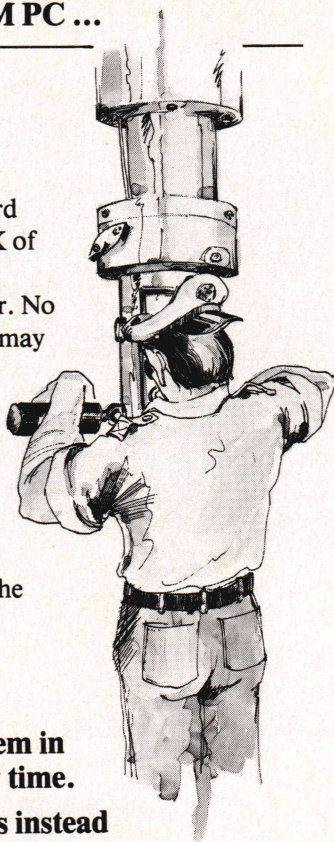
The Periscope system is \$295. It carries a 30-day money-back guarantee and includes the memory board, remote break-out switch, debugger software, 100-page manual, and quick-reference card. The memory board is warranted for one year. A demonstration disk is \$5.00.

System requirements for Periscope are an IBM PC, XT or Compaq, PC-DOS, 64K RAM, 1 disk drive and an 80-column monitor. For MasterCard and Visa orders only, call 800/421-5300 (ext. R96) 24 hours a day. For additional information, call 404/256-3860 from 9 AM to 5 PM Eastern Time.

Get your programs up and running;

UP PERISCOPE!

Data Base Decisions / 14 Bonnie Lane / Atlanta, GA 30328



error if you attempt to convert a real value of -32768 (8000H). Since this is a legal integer value, and right in the middle of the range of hex numbers, I think this has to be called a bug.

Listing Four (page 14) shows one way of programming around this problem. It works just fine when using real numbers and integers to represent hex numbers between 0 and FFFFH, but it would not be very useful in an application that requires both positive and negative real numbers.

Sincerely,
William Ames
8720 Topanga Cyn. Blvd. #8
Canoga Park, CA 91304

Borland assures us that this bug will be fixed in Version 3.0 of Turbo Pascal.—Ed.

Fast Hartley Transform

Dear DDJ,

We were pleased to see Ron Ullmans's letter in the December 1984, issue of *Dr. Dobb's Journal* regarding the Hartley Transform and the accompanying listing of a software program. We, too, consider its advantage to be great.

Your readers may wish to know Stanford University has proprietary rights on this technology, developed by Professor Ronald N. Bracewell. Patent rights are pending.

Use of the software without a license may be an infringement of those proprietary rights. However, a license under reasonable terms to the technology is available from Stanford.

Incidentally, we have called this development the "Bracewell" transform, in recognition of Professor Bracewell's fundamental contribution.

Sincerely,
Lisa Kuuttila
Consulting Associate
Office of Technology
Licensing
Stanford University

DDJ

Listing One

```
; UPDATE PATCHES FOR PUBLIC.ASM

vers    equ    1$2                ;update PUBLIC.ASM version number

; fix 1.  Insert the following code in the 'SAHEXT' routine
;         immediately preceeding its 'ret' instruction.

        rnz          ; v 1.1
        inx    h      ; extent is 0, check overflow (s2) ext.
        inx    h
        mov    a,m
        ani    7Fh
;
        ret          ; end of SAHEXT routine

; fix 2.  Insert the following macro statement at the label 'xit:'
;
xit:     dobdos    dmafn,80h        ;restore default dma for SUBMIT under ccp.
        lspd      ustack          ; v 1.2
        ret
```

End Listing One

Listing Two

TURBO PASCAL Program Lister, Copyright 1983 Borland International
Listing of: STATIC.PAS

```
program TestStaticVariables;
{ This program demonstrates that the "typed constants" of Turbo Pascal }
{ are really static variables. Each time the main loop calls ShowStatic }
{ the typed constant I is incremented by 1. When ShowStatic is called }
{ again the incremented value is written to the CRT. }
var
  Ch : Char;
  I  : integer;

procedure ShowStatic;
{ This prints and then increments the "constant" I }
const
  I : integer = 0;
begin
  Writeln(I);
  I := I + 1;
end;

function UseHeap(L : integer) : integer;
{ This is included to demonstrate that the typed constant above is not }
{ destroyed when another procedure uses the heap or stack. }
var
  I : integer;
  J : ^integer;
begin
  I := -L;
  new(J);
  J^ := L;
  L := L + 1;
  if L < 5 then
    L := UseHeap(L);
  Dispose(J);
  UseHeap := L;
end;
```

(Continued on next page)

Listing Two

```
begin
  repeat
    read(Kbd, Ch);
    I := UseHeap(0);
    ShowStatic;
    Until Ch = #13;
  end.
```

End Listing Two

Listing Three

TURBO PASCAL Program Lister, Copyright 1983 Borland International
Listing of: CNVRT1.PAS

```
program RealToInteger1;
{ This program can be used to demonstrate a problem in }
{ converting real numbers to integers. }
var
  Nmbr : real;
  I     : integer;

begin
  Readln(Nmbr);
  { if Nmbr = 32768.0 the following produces a runtime error }
  if (Nmbr > 65535.0) or (Nmbr < 0) then
    { I am only interested in the range of 4 digit Hex numbers }
    Writeln(' overflow error')
  else begin
    if Nmbr < 32768.0 then
      I := Trunc(Nmbr)
    else
      I := Trunc(Nmbr - 65536.0);
    Writeln(I);
    end;
  end.
```

End Listing Three

Listing Four

TURBO PASCAL Program Lister, Copyright 1983 Borland International
Listing of: CNVRT2.PAS

```
program RealToInteger2;
{ this shows one way around the bug demonstrated in version 1 }
var
  Nmbr : real;
  I     : integer;

begin
  Readln(Nmbr);
  { 32768.0 is converted correctly by the following code }
  if (Nmbr > 65535.0) or (Nmbr < 0) then
    { I am only interested in the range of 4 digit Hex numbers }
    Writeln(' overflow error')
  else begin
    if Nmbr < 32768.0 then
      I := Trunc(Nmbr)
    else
      begin
        I := Trunc(Nmbr - 65535.0);
        I := I - 1;
      end;
    Writeln(I);
    end;
  end.
```

End Listing Four

**The Macintosh
Software
Library
just got fatter . . .**

**It's about
TIME.
AND SPEED.**

FASTER COMPUTER OPERATION with disk operations 2 to 5 times faster.

QUICK PROGRAM TRANSFER FROM OTHER CP/M COMPUTERS using built-in transfer utilities and standard CP/M file structure.

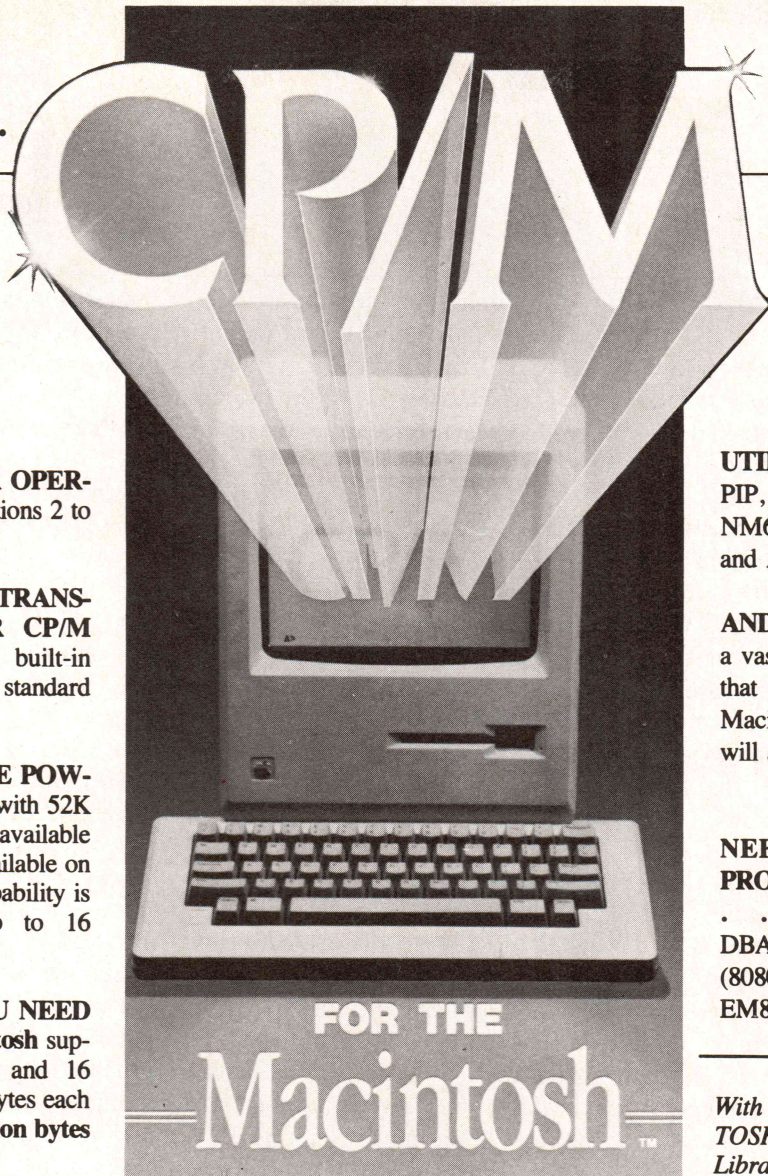
RUNS BIGGER MORE POWERFUL PROGRAMS with 52K transient program area available on 128K Mac, 300K available on 512K system and the capability is built-in to manage up to 16 Megabytes of RAM.

PLUG-IN WHAT YOU NEED ...CP/M for the Macintosh supports up to 2 printers and 16 Drives of 512,000,000 bytes each for a maximum of 8 Billion bytes of storage.

MORE DISK SPACE AVAILABLE ... each disk drive has 360 bytes available, often eliminating the need for additional drives.

CHOOSE ANY PRINTER letter quality or dot matrix ... any serial printer will work just fine.

MEDIA PROBLEMS?...NONE ... all that is needed to transfer programs from 5¼ to 3½ is a modem transfer program to operate with our Modem 7 Communications Compatible Program ... this enables programs to be transferred from **XEROX, OSBORNE, HEATHKIT, KAYPRO, RADIO SHACK** and others.



UNLIMITED EXPANDABILITY is made possible with the total and easy control of serial expansion ports which allows many forms of plug-ins.

UTILITIES ... such as STAT, PIP, DDT, ED, LINK68, INIT, NM68, SIZE, LO68, RELOC, and AS68.

AND We are currently compiling a vast Library of CP/M software that can be ported over to the Macintosh and our Macrolibrary will also be available soon.

NEED POPULAR WORD PROCESSING PROGRAMS? . . . Run **WORDSTAR®** **DBASEII®** **AND** other CP/M (8080) 2.2 Programs with our EM80 Emulator . . **ONLY \$195.!**

With CP/M FOR THE MACINTOSH the Macintosh Software Library just got fatter!

THE 10 FEATURES:

1. 6 Disk System
2. Documentation from Digital Research and I.Q.
3. MacroAssembler (*no extra charge*)
4. Modem 7 Compatible Communications Transfer Program (*no extra charge*)
5. C Compiler (*no extra charge*)
6. Text Editor (*no extra charge*)
7. Menu Program (puts menus on your programs) (*no extra charge*)
8. Standard Printer Driver (*no extra charge*)
9. Copy Program (*no extra charge*)
10. Leir-Seigler ADM3A Terminal Emulation Program (*no extra charge*)

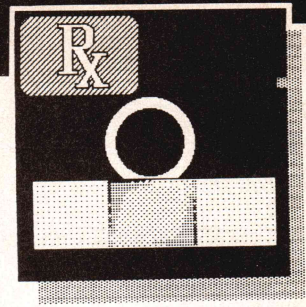
**All This For Only
\$395 Retail**

Call or write for your Macintosh CP/M Software Catalog.



I.Q. Software
2229 E. Loop 820 N.
Ft. Worth, Texas 76118
(817) 589-2000

Macintosh is a registered trademark of Apple Computer, Inc. CP/M is a registered trademark of Digital Research, Inc. WORDSTAR is a registered trademark of Micropro. DBASE II is a registered trademark of Ashton-Tate. ©1985 I.Q. Software



by D. E. Cortesi

Welcome, Microans

It seems that *DDJ* has picked up a batch of new readers, the erstwhile subscribers to the late, lamented *Micro*. Let's recap the purpose of this column for their benefit. It's supposed to be "a place for the display of techniques and discoveries." That's what we said the first time it appeared, in May 1981. Also, "We'd like to tell of unobvious uses for standard utilities. We want to uncover errors in documentation, to warn people away from pitfalls, and to show off those 'Eureka!' moments that make systems work rewarding."

For a long time, most of the contributions to the Clinic involved CP/M and the Z80. Nowadays, we hear more from MSDOS/8086 users. Well, stand back, Zilog lovers; make room, Intel-lects! Now we have among us people who understand *real* computers like the 6502, the 6809, and the 68000 and *real* operating systems like OS9, SOS, and PRODOS. We can barely wait for the neat contributions they'll be sending in. Soon.

Wood Blocks

As bait for the *Micro* readers (hey, folks, we really like reader contributions here), we present our wood blocks program for the Atari. It's a little thing (see Listing at right) that we ran up one day while experimenting with the start, select, and option keys. These three keys control the three lowest bits of a byte in the Atari address space. The program samples that byte, and each time the combination of operated keys changes, it emits a heavily damped *klonk* at a different pitch. Play it by mashing down all three keys and then sort of

working your fingers around.

The simulation of damped tones is pretty good. Experiment with the damping factor and with the expression that generates a note value from the key combination.

Roll Over, Keyboard

We've an interesting letter here from Charles Davis, who lives in a town with the delightful name of Flower Mound, Texas. Besides submitting some throughput timings for an Atari system (at last!), he's been studying keyboards:

"Perhaps you could survey another area of concern. I am either a very good typist or a very sloppy one! I need a keyboard with what used to be called 'N-key rollover.' These are hard to find. The current generation of computers and terminals does not seem to supply this feature. Most marketing and sales people ... are unaware of the feature and therefore the sales literature does not mention it, even when it is present.

"A survey of the popular computers and terminals with reference to rollover would be interesting to me and, I

think, to many of your readers. My procedure for checking rollover is to depress several keys in sequence, holding each until all are depressed, then I lift them all in the same order. I observe what happens on the screen. There seem to be three types of response:

- the first N keys (type 1)
- the last N keys (type 2)
- only the first and last keys (type 3)

The major variable is how big N is. Rarely must I find a helper with more fingers. One terminal manufacturer admitted that their terminals had N-key rollover and that N was supposed to be 5, but some key sequences did not work right so they would only guarantee N to be 3. This points up the weakness of my procedure: I do not do an exhaustive check and some problem characters may be missed. Without knowledge of the key matrix and the scanning algorithm, it would take too long to find the problems."

Davis sent along the results from some terminals he'd tested (see Table 1 on page 17). However, our first at-

```

10 REM WOODBLOX FOR ATARI, D.E. CORTESI
20 KEYBYTE=53279
   : DAMPING=0.29
30 SAMPLE=PEEK(KEYBYTE)
   : IF SAMPLE=BUTTONS THEN GOTO 30
40 BUTTONS=SAMPLE
   : PITCH=12*(8-SAMPLE)
   : VOLUME=15
50 SOUND 0,PITCH,12,VOLUME
   : VOLUME=VOLUME-VOLUME*DAMPING
   : IF VOLUME >= 1 THEN GOTO 50
60 SOUND 0,0,0,0
   : GOTO 30
    
```

Terminal	Response	Rollover
ADDS Regent 20	type 1	N > 10
ADDS Viewpoint	type 1	N > 10
Appollo DN600 Workstation	type 1	N > 10
Atari computers	type 3	n.a.
CIT 101E	type 1	N = 2
Daisy Logician Workstation	type 1	N > 10
Hazeltine 1410	type 3	n.a.
Heath/Zenith 19 and 29	type 4	N = 2
HP-85	type 3	n.a.
HP-9816	type 1	N > 10
IBM PC/XT	type 1	N > 10
Perkin-Elmer Bantam 550	type 1	N = 3
Soroc IQ-120	type 3	n.a.
Televideo 950	type 1	N = 2
Wyse 50	type 1	N = 3

Table 1

Charles Davis' survey of the number of keys that can be "rolled over" on different terminals. Types are: 1, first N keys; 2, last N keys; 3, first and last only; 4, first N keys then garbage.

Command Code	Type of Driver	Operation
0	either	initialize
1	block	media (sic) check
2	block	build Bios Parameter Block
3	either	IOCTL input call
4	either	input
5	character	nondestructive input no wait
6	character	input status check
7	character	input flush
8	either	output
9	either	output with verify
10	character	output status
11	character	output flush
12	either	IOCTL output call
13	either	notification of file-open
14	either	notification of file-close
15	either	IOCTL query: removable disk?

Table 3

The command codes to which an MSDOS device driver must respond, omitted from IBM's DOS *Technical Reference* manuals. Codes 13, 14 and 15 are new with PC DOS 3.0.—Codes 3, 12 and 15 are passed only to drivers that indicate IOCTL support in their Device Header Attribute word.

Demo Disk Call (800) 327-5895

VEDIT PLUS

The First Multiple File Programmable Editor

VEDIT has been acclaimed for the last five years as the Industry Standard in text editing.

InfoWorld Software Report Card

VEDIT 1.36

	Poor	Fair	Good	Excellent
Performance	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Documentation	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Ease of Use	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Error Handling	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Now there's VEDIT PLUS.

VEDIT PLUS is the ideal tool for program editing and technical writing. It gives you every editing function you expect, plus it can:

- Edit multiple files of any size
- Compare files
- Be fully customized
- Perform arithmetic computations
- Be expanded with print formatting & spell checking-correcting

The power of VEDIT PLUS lets you increase your productivity with:

- Numeric, relational, and logical functions
- If-Then-Else decision making
- Easy file handling
- The ability to create prompts for user input and menus
- Special programming features

Expect a lot from VEDIT PLUS. It's small (23K), fast and it's from CompuView - nationally recognized for user support.

VEDIT PLUS\$225
VEDIT\$150

CompuView

PRODUCTS, INC.

1955 Pauline, Ann Arbor, MI 48103
(313) 996-1299 - (800) 327-5895

AT LAST
S-100 ↔ 488
THAT
DOES
EVERYTHING
YOU WANT
IT TO DO



D&W DIGITAL, INC.
20655 Hathaway Avenue
Hayward, California 94541
(415) 887-5711

tempt to test a keyboard produced a new result that you should watch out for:

- the first N keys plus a garbage key (type 4)

This is the H19 problem we reported here in August 1982; it results from a lack of blocking diodes in the key matrix. Heath hasn't learned anything since, either. On our Heath/Zenith 29, holding L, I, and then S produces "LIQS" on the screen. We have to be very precise typing the word "list" or it comes out "liqst." Worse, some combinations produce false function keys. The combination E, I, then P generates a false down-arrow (escape-B) sequence, for instance. So the H/Z29 is an N=2 machine, even though for some key combinations it will pass Davis' test as a type 1 with much higher N. Incidentally, the good combinations occur on the home row. That suggests that you might apply Davis' test on the top or bottom rows.

Charting DOS

We've been boning up on MSDOS and PCDOS lately, in preparation for writing a book and we've compiled the chart of DOS services that appears in Table 2 (page 19) as a study aid. For each possible Int 21 service it shows the request number in decimal and hex, the releases that support it, and a capsule description of the service. The services are organized into groups by function. The groups and the services within the groups are generally in ascending order by request number, but functional relationships are given precedence.

The idea is that you can look for the service you need by scanning a small group of related ones. The capsule descriptions are just enough to differentiate between similar functions. Having found the service you need, you can access its detailed description in your DOS manual through its request number.

Our study of PCDOS has been impeded no small amount by the poor quality of the IBM DOS manuals, especially those for DOS 3.0. For one

example, the documentation on service 56/38h (get or set country-dependent information) is so badly done that we honestly cannot make sense of it. If you are a reader who can write a comprehensible, accurate description of the DOS 3 version of service 38h, please do so and send it to the Clinic.

For another example, the chapter on device drivers omits any specification of the command codes to which a device driver is supposed to respond. In the DOS 2.1 manual you could still cull the essential information from the comments of a sample program. In DOS 3.0, the sample program has been dropped from the manual. The book says just enough that you can infer that command codes 3, 4, 8, 9, and 12 specify input or output, but there is no way you can tell which code calls for what action!

Fortunately, the missing information can be found in Microsoft's version of the technical manual, from which we constructed Table 3 (page 17). Although some command codes are specific to character or block devices, it seems wisest for all drivers to expect all command codes. They can treat the unwanted ones as null operations, or they can reject them with the "unknown command" error code.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 190.

Service number in decimal and hex

Appears in which releases

Functional group

Capsule description

0/00	1/2/3	program cntrl	end program (requires PSP *CS)
49/31	-/2/3	program cntrl	end, stay resident, AL=ret, DX=size
76/4C	-/2/3	program cntrl	end program, AL=ret
38/26	1/2/3	program cntrl	make daughter PSP at (DX:00)
75/4B	-/2/3	program cntrl	(AL=0) load, run daughter from path *DS:DX
77/4D	-/2/3	program cntrl	get AH=term code, AL=ret of daughter
75/4B	-/2/3	program cntrl	(AL=3) load DS:DX —> path as overlay
98/62	-/-/3	program cntrl	get BX=paraddr of my PSP
48/30	-/2/3	system info	get DOS version number
42/2A	1/+ /3	system info	get CX=year, DH=mo, DL=day (2,3:AL=dow)
44/2C	1/2/3	system info	get time as CH:CL:DH.DL
51/33	-/2/3	system info	(AL=00) get Break switch in DL
53/35	-/2/3	system info	get int(AL) vector to ES+BX
56/38	-/2/+	system info	(AL=00) get country-dependent info
72/48	-/2/3	system info	(BX=FFFF) get size available RAM to BX
89/59	-/-/3	system info	get extended error to AX, BH, BL, CH
37/25	1/2/3	system control	set int(AL) vector to (DS+DX)
43/2B	1/2/3	system control	set date from CX, DX
45/2D	1/2/3	system control	set time from CH:CL:DH.DL
51/33	-/2/3	system control	(AL=01) set Break switch from DL.0
56/38	-/-/3	system control	(??) set country-dependent info
72/48	-/2/3	system control	allocate BX paragraphs
73/49	-/2/3	system control	return RAM from ES=paraddr
74/4A	-/2/3	system control	resize ES=paraddr to BX paras
1/01	1/2/3	kbd/handle 0	get key: wait, echo, break-check
6/06	1/2/3	kbd/handle 0	(DL=FF) get key: noecho, nowait, nobreak
7/07	1/2/3	kbd/handle 0	get key: wait, noecho, nobreak
8/08	1/2/3	kbd/handle 0	get key: wait, noecho, break
10/0A	1/2/3	kbd/handle 0	buffered line input with editing
11/0B	1/2/3	kbd/handle 0	test if key available (break)
12/0C	1/2/3	kbd/handle 0	flush buffered keys, then do (AL)
2/02	1/2/3	scrn/handle 1	put DL with editing and break-check
6/06	1/2/3	scrn/handle 1	(DL < > FF) put DL, noedit, nobreak
9/09	1/2/3	scrn/handle 1	iterate DOA 2 over *DS:DX to \$
3/03	1/2/3	aux/handle 3	get byte from COM1
4/04	1/2/3	aux/handle 3	put byte to COM1
5/05	1/2/3	prt/handle 4	put byte to LPT1
25/19	1/2/3	disk info	get AL=default drive
47/2F	-/2/3	disk info	get Disk Transfer Address to ES:BX
27/1B	1/+ /3	disk info	get FAT *DS:BX (2,3: only 1st byte of FAT) . .
28/1C	-/2/3	disk info	. . and CX=bytes/sec AL=sec/unit DX=units/disk
54/36	-/2/3	disk info	BX=free units, CX/DX/AL as for 1C
84/54	-/2/3	disk info	get write-verify switch to AL.0

Table 2
Chart of DOS Services

(Continued on next page)

(Continued)

13/0D	1/2/3	disk control	reset disk system
14/0E	1/2/3	disk control	select drive, return count of drives
26/1A	1/2/3	disk control	set Disk Transfer Address = DS:DX
46/2E	-/2/3	disk control	set write-verify switch from AL.0
15/0F	1/2/3	fcf control	open file from FCB
16/10	1/2/3	fcf control	close file from FCB
17/11	1/2/3	fcf control	scan current directory for first match
18/12	1/2/3	fcf control	scan current directory for next match
19/13	1/2/3	fcf control	erase files matching FCB
22/16	1/2/3	fcf control	make new file from FCB
23/17	1/2/3	fcf control	rename file from FCB
35/23	1/2/3	fcf control	get size of file to fcb
36/24	1/2/3	fcf control	set relrecno from current file position
40/28	1/2/3	fcf control	(CX = 0) trunc or stretch file to size
41/29	1/+/3	fcf control	parse filespec *DS:SI into fcb *ES:DI
20/14	1/2/3	fcf i/o	read next record to DTA
21/15	1/2/3	fcf i/o	write DTA to next record
33/21	1/2/3	fcf i/o	read record at direct address
34/22	1/2/3	fcf i/o	write record to direct address
39/27	1/2/3	fcf i/o	block read CX records to DTA
40/28	1/2/3	fcf i/o	(CX > 0) block write CX records from DTA
57/39	-/2/3	directories	make directory for path *DS:DX
58/3A	-/2/3	directories	erase directory at end of path *DS:DX
59/3B	-/2/3	directories	set current path as path *DS:DX
71/47	-/2/3	directories	get current path of DL=drv to DS:DI
63/3F	-/2/3	file i/o	read CX bytes from BX=handle to DS:DX
64/40	-/2/3	file i/o	write CX bytes to BX=handle from DS:DX
60/3C	-/2/3	file control	create/trunc file (path *DS:DX, CX=attr)
91/5B	-/-/3	file control	create new or fail (path *DS:DX, CX=attr)
90/5A	-/-/3	file control	create unique file (path *DS:DX, CX=attr)
61/3D	-/2/+	file control	open file (path *DS:DX, AL=access type)
62/3E	-/2/3	file control	close file (BX=handle)
65/41	-/2/3	file control	erase file (path *DS:DX)
66/42	-/2/3	file control	seek (BX=handle, AL=method, CX+DX=offset)
67/43	-/2/3	file control	(AL=0) get CX=attr of file path *DS:DX
67/43	-/2/3	file control	(AL=1) set attr of file path *DS:DX to CX
69/45	-/2/3	file control	create a duplicate file handle
70/46	-/2/3	file control	force a handle to be a duplicate
78/4E	-/2/3	file control	search 1st match to path *DS:DX
79/4F	-/2/3	file control	search next match to path *DS:DX
86/56	-/2/3	file control	rename/move path *DS:DX to path *ES:DI
87/57	-/2/3	file control	(AL=0) get CX=time, DX=date of path *DS:DX
87/57	-/2/3	file control	(AL=1) set CX=time, DX=date of path *DS:DX
92/5C	-/-/3	file control	(AL=0) at CX+DX in BX=hdl, lock SI+DI bytes
92/5C	-/-/3	file control	(AL=1) release bytes locked by prior call

Table 2
Chart of DOS Services

THE PROGRAMMER'S SHOP™

helps compare, evaluate, find products. Straight answers for serious programmers.

SERVICES

- Programmer's Referral List
- Compare Products
- Help find a Publisher
- Evaluation Literature free
- BULLETIN BOARD - 7 PM to 7 AM 617-826-4086
- Dealer's Inquire
- Newsletter
- Rush Order
- Over 700 products

Free Literature - Compare Products

Evaluate products Compare competitors. Learn about new alternatives. One free call brings information on just about any programming need. Ask for any "Packet" or "Addon Packet" ☐ ADA, Modula ☐ "AI" ☐ BASIC ☐ "C" ☐ COBOL ☐ Editors ☐ FORTH ☐ FORTRAN ☐ PASCAL ☐ UNIX/PC or ☐ Debuggers, Linkers, etc.

RECENT DISCOVERIES

FASTER C - Lattice users eliminate Link step. Normal 27 seconds, Faster C in 13 secs. MSDOS \$95

ARTIFICIAL INTELLIGENCE

EXSYS - Expert System building tool. Full RAM, Probability, Why, Intriguing, serious. PC DOS \$275

GC LISP - "COMMON LISP", Help, tutorial, co-routines, compiled functions, thorough. PC DOS \$455

IQ LISP - MACLISP & INTERLISP. Full RAM. Liked. PC DOS \$155

TLC LISP - "LISP-machine"-like, all RAM, classes, turtle graphics 8087. CP/M-86, MSDOS \$235

INSIGHT 1 - Expert Sys. Dev't, decent PC DOS \$95

PROLOG-86 - Learn fast, Standard, tutorials, samples of Natural Language, Exp. Sys. MSDOS \$125

Expert System front-ends for PROLOG: APES (\$275), ES/P (\$1895)

Other solid alternatives include: MuLISP-86 (\$189), WALTZ LISP for CPM (\$159), MicroPROLOG (\$275)

EDITORS FOR PROGRAMMING

BRIEF Programmer's Editor - undo, windows, reconfigurable, macro programs, powerful. PC DOS \$195

VEDIT - well liked, macros, buffers, CPM-80-86, MSDOS, PC DOS \$119

MACINTOSH

We evaluate, carry every available programmers product. Ask.

C LANGUAGE

INSTANT C - Interactive development - Edit, Source Debug, run. Edit to Run - 3 Secs. MSDOS \$495

"INTRODUCING C" - Interactive C to learn fast. 500 page tutorial, examples, graphics. PC DOS \$95

MEGAMAX C - native Macintosh has fast compile, tight code, K&R, toolkit, .OBJ, DisASM MAC \$275

Audio-based C tutorials. Overview \$95. Full \$295

CLIBRARIES

COMMUNICATIONS by Greenleaf (\$159) or Software horizons (\$139) includes Modem7, interrupts, etc. Source. Ask for Greenleaf demo.

C SHARP Realtime Toolkit - well supported, thorough, portable, objects, state sys. Source MANY \$600

APPLICATION TOOLKIT by Shaw - Complete: ISAM, Screen, Overlay mgmt, report gen, Strings, String math. Source. CPM, MSDOS \$495

ROMPack - special \$Main .EXE editor, source, tech support, 8086. \$185

DEBUGGERS

PERISCOPE DEBUGGER - load after "bombs", symbolic, "Reset box", 2 Screen, own 16K. PC DOS \$279

SOURCE PROBE by Atron for Lattice, MS C, Pascal. Windows single step, 2 screen, log file. \$395

FORTRAN LANGUAGE

MacFORTRAN - full '77, '66 option, toolbox, debugger, 128K or 512K, ASM-out option MAC \$375

DR/Fortran-77 - full ANSI 77, 8087, overlay, full RAM, big arrays, complex NUMS., CPM86, MSDOS \$249

Ask about Microsoft, Supersoft, others.

OTHER LANGUAGES

ASSEMBLER - ask about FASM-86 (\$95), ED/ASM (\$100) - both are fast, compatible, or MASM (\$125), improvements.

BetterBASIC all RAM, modules, structure. BASICA - like \$185

HS/FORTH - '79 & '83 Standards, full RAM, ASM, BIOS, interrupts, graph, multi-task, optimizer MSDOS \$250

MBP COBOL has screen control, strong doc, '74 interm., fast. MSDOS \$680

SUPPORT PRODUCTS

BASIC DEVELOPMENT SYSTEM - (BDS) for BASICA; Adds Renum, crossref, compress. PC DOS \$115

PLINK-86 for Overlays, most lang., segment control. MSDOS \$325

ProYAM Communications Package - All a programmer'd want. TTY, VT 100, 3101, MODEM7, BBS. Remote, macros, windows MSDOS \$139

CODESMITH - visual, interactive debugger. Symbolize, modify code \$129

"C" LANGUAGE

	OUR PRICE
MSDOS C86-8087, reliable	call
Instant C - Inter., fast, full	495
Lattice 2.1 - improved	call
Microsoft C 2 x	279
Williams, debugger, fast	call
C Systems & debugger	175
CPM80: EcoPlus C - faster, SLR	275
BDS C - solid value	125
MACINTOSH: Softworks	365
Megamax-object, full	275
Consular's MAC C	275
Compare, evaluate, consider other Cs	

BASIC

	RUNS ON
Active Trace-debug	86/80 75
BASCOM-86 - MicroSoft	8086 279
BASIC Dev't System	PC DOS 115
BetterBASIC - 640K	PC DOS 185
CB-86 - DRI	CPM86 419
Prof. BASIC Compiler	PC DOS 89
Databurst - screens	MSDOS 215
SCREEN SCULPTOR	PC DOS 115

Ask about ISAM, other addons for BASIC

SERVICE

ALL PRODUCTS - We carry 700 products for MSDOS, CP M 86, CP M 80, Mac-Intosh and key products for other micros.

EDITORS Programming

	RUNS ON	OUR PRICE
BRIEF - Intuitive, flexible	PC DOS	195
C Screen with source	86/80	75
Epsilon - like EMACS	PC DOS	195
FINAL WORD-for manuals	86/80	215
MINCE-like EMACS	PC/80	149
PMATE-powerful	8086	185
VEDIT-full, liked	86/80	119

UNIX PC

COHERENT - for "C" users PClike 475
COHERENT-NCI-Realtime PClike call
XENIX - plus C to MSDOS PC 1275
Ask about run-times, applications, DOS compatibility, other alternatives. UNIX is a trademark of Bell Labs

LANGUAGE LIBRARIES

GRAPHICS: Graphic-source in C	MSDOS	219
GRAPHMATIC-3D, FTN, PAS	PC DOS	125
HALO-fast, full-all lang.	PC DOS	139
FILE MGMT: BTrieve-all lang.	MSDOS	215
CIndex - source, no royal.	86/80	369
Ctree-source, no royal.	ALL	369
dBc ISAM by Lattice	8086	229
db VISTA - "Network" Structure	MSDOS	465
PHACT-up under UNIX, addons	MSDOS	225
OTHER: Cutil by Essential	MSDOS	129
Greenleaf - 200 +	MSDOS	159
CSharp - Real-Time	MSDOS	600
PORTABLE C to PC, Mac, II	Many	125
SOFT Horizons - Blocks I	PC DOS	139
SCREEN: CURSES by Lattice	PC DOS	125
CView - input, validate	PC DOS	195
MetaWINDOW - icons, clip	PC DOS	139
PANEL - many lang, term	MSDOS	249
ProScreen - windows, source	PC DOS	415
Windows for C	MSDOS	175

FORTRAN

	RUNS ON	OUR PRICE
MS FORTRAN-86 - Implr.	MSDOS	\$ 239
DR Fortran-86 - full '77'	8086	249
PolyFORTRAN-XREF, Xtract	PC DOS	165

OTHER PRODUCTS

Assembler & Tools - DRI	8086	159
Atron Debugger for Lattice	PC DOS	395
C English - dBase to C	MSDOS	750
C Helper: DIFF, xref, more	86/80	135
CODESMITH-86 - debug	PC DOS	129
MacASM-full, fast, tools	MAC	115
MBP Cobol-86 - fast	8086	680
Modula 2 for	MAC, PC DOS	90
Micro: SubMATH-FORTRAN full	86/80	250
Microsoft MASM-86	MSDOS	125
MSD Debugger	PC DOS	119
Multilink - Multitasking	PC DOS	265
PC FORTH - well liked	MSDOS	219
PFIX-86 Debugger	MSDOS	169
PL 1-86	8086	495
Polylibrarian - thorough	MSDOS	95
PolyMAKE	PC DOS	95
PROFILER by DWB - flexible	MSDOS	109
Prolog-86-Learn, Experiment	MSDOS	125
SLK F - Copy Protection	PC DOS	145
SYMD debugger-symbols	PC DOS	119
TRACE86 debugger ASM	MSDOS	115

Note: All prices subject to change without notice. Mention this ad. Some prices are specials. Ask about COD and PDs. All formats available

Call for a catalog, literature, and solid value

800-421-8006

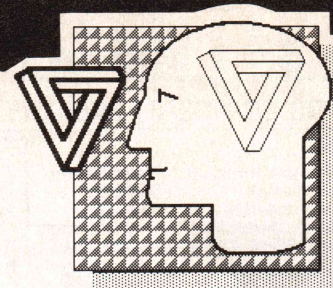
THE PROGRAMMER'S SHOP™

128-1 Rockland Street, Hanover, MA 02339

Visa MasterCard 8517
Mass: 800-442-8070 or 617-826-7531

Dr. Dobb is a Subversive

by D. E. Cortesi



The publishers of DDJ recently threw a party to celebrate the 100th issue of the magazine. It was a friendly affair, with lots of sincere speeches about how important DDJ has been and about how it still has a great role to play. There was a lot of talk about "hackers:" what they accomplished; where DDJ stood in relation to them; where they might have gone now.

they were expressed, the sentiments seemed incomplete to me. Something was being overlooked, but I couldn't figure out what it was until I was halfway home. Then I lay awake working out the speech that ought to have been given, but wasn't. And this is it . . .

We've heard a lot about "hackers" in the last few minutes. I would like to protest the use of that word. I understand why it's being used. It's because nobody can think of the right word to describe the people who founded *DDJ* and the people who read and profited from it in its earliest days. We know they weren't part of the establishment; we know they didn't play a conventional role in the society of their day. We have, perhaps, an uneasy feeling that we aren't quite like them and that they wouldn't find us very interesting—even though perhaps we *were* them just a few years ago.

So we scratch around in our vocabularies, searching for the right word to describe these departed people, and the best we can come up with is "hacker." Well, it's the wrong word.

The right word is . . . *revolutionary. Subversive. Bomb-throwing anarchist.*

To me, the hacker state is a frame of mind in which one is turned inward on oneself and the machine: one's

spiritual third eye rolls up into the head. The people who birthed *DDJ* and personal computing in general were quite the opposite. Their eyes were turned outward on society—and twinkled with mischief.

They were bent, quite consciously, on overthrowing the established order. It was to be a nonviolent overthrow, one conducted if possible with whimsy and good humor, but it was to be an overthrow, a toppling, nonetheless.

The central tenet of this jolly revolution was (and still is) the belief that *if we put enough computers in the hands of enough people, wonderful changes will take place.* What the wonderful changes are . . . well, that varies with the year and the person you're talking to. But that they will be wonderful, as well as radical in the deepest and best sense of the word, no computer subversive doubts. I certainly don't.

The founder of the movement and its only theorist, its Marx or Che Guevara if you like, is Ted Nelson. His book, *Computer Lib*, was written in reaction to computers, their employers, and their professional priesthood as those things existed in the early 70s. It was a protest: "Either computer systems are going to go on inconveniencing our lives, or they are going to be turned around to make life better," and then, in caps, a battle cry: "COMPUTER POWER TO THE PEOPLE! DOWN WITH CYBERCRUD!"

And to make sure we didn't miss the point, there was the symbol of the clenched fist on the cover. I hope I'm not the only one here who remembers when the word *Lib* and the upraised fist were potent, dangerous symbols. That Nelson borrowed both for his manifesto indicates how serious he was.

I don't know if the People's Com-

puter Company admitted to subversive intentions in public or not. Unfortunately for me, I was out of the country during its glory days. I was aware of micros and an early subscriber to *DDJ*, but I was only a fellow traveler to the revolution, not a revolutionary myself.

But whether the intentions were admitted or not, I'm positive that they were there. No one who's spent any time with Bob Albrecht, or with Ramon Zamorra, who founded ComputerTown USA!, could have any doubt that these people are motivated by a deep love of mischief. Nothing pleases them better than a well-toppled apple cart or a well-rocked boat.

Another revolution that I missed out on was the birth of the acid culture in the early 60s. But I suspect that the spirit of People's Computer Company had a lot in common with the spirit of Ken Kesey's Merry Pranksters, and with that of the Diggers and the various communards of the 60s. They all felt that they'd stolen the fire of the gods and that they'd damn well better set as many blazes as they could before the gods came to take it back.

After all, what could possibly be more subversive than training children to operate the tools of the establishment, *without* at the same time inculcating the establishment's line of thinking? But that was what Albrecht, Allison, *et al.*, were doing. I don't know if this is what was specifically in their minds or not. But consider. You have all had the following experience. You program something in BASIC; you find that the program doesn't work because of the inevitable bugs; then, after you pick them off, you find that it *still* doesn't work because the limited accuracy of binary floating point has turned your nu-

merical results into oatmeal. Nobody who has been through that can ever look at their bank statement in quite the same way again. Or their phone bill. Or, if there is a live connection between one side of their brain and the other, can they ever feel quite the same about a proposal for an antibalistic missile system as they might once have done before their personal encounter with computers in the raw.

Albrecht and company used time-sharing terminals to start indoctrinating children and susceptible adults, but then the micro came along. The earliest micros were pitiful, but to the eyes of a computer subversive they appeared to be the sword of liberation. A computer you could hold in your hand!

For a parallel, try to imagine that Gutenberg had never invented his printing press; that the only way for books to be reproduced was for them to be handwritten by scribes. Of course, only the wealthy and the government can afford to pay scribes and buy vellum, so the only thoughts that circulate in book form are the ones acceptable to the powers that be. Then somebody invents the typewriter: a cheaper, faster way of being a scribe, one that anybody can learn. So you start a counterculture book factory, teaching people to use typewriters to duplicate unapproved literary works.

And then somebody walks in the door with the latest invention, *the Xerox copier*. And just that simply, your revolution has been won; now it's just a matter of mopping up.

In the case of the computer revolution, the mopping up is still going on. The micro *has*, in fact, hurt the computer bureaucrats, the people who would try to make you swallow an inconvenience on the grounds that the computer wouldn't let them be helpful. These days they find it a lot harder to say "the computer won't let me" because so many clients are likely to answer "Why not? My computer would." There really is a lot less these days of what Nelson called cybercrud, and we can credit that to the wide popular exposure to micros.


But there is a lot more to be done. We've spread computers all over the

place and still have no millenium, nor anything remotely like it. There's been no great liberation of minds or spirits. The revolution has been assimilated, just as the automotive, sexual, drug, and free speech revolutions were assimilated before it, and the feminist revolution is being assimilated now: they come, they pass, things are different but not very much so. Surely by now the wonderful thing should have happened. Surely by now somebody should have written the program that will set us all free? But no.

One reason for this failure is that

the computer did not turn out to be a sword that could be placed in anyone's hand. It isn't a sword at all, nor anything like a simple edged or pointed weapon. For complexity it's more like a whole battalion of soldiers. If you tell some poor shmuck to start wielding one, you put him in the position of a private promoted instantly to general and asked to take over management of a battle.

To change metaphors nimbly in midstream, we have made the belated discovery that the computer priesthood that we deposed was not entirely a group of fakes; they had more



*C Programmers
Quit Working
So Hard!*

THE GREENLEAF FUNCTIONS™

The GREENLEAF FUNCTIONS GENERAL LIBRARY has over 200 functions in C and assembler. Strength in DOS, video, string, printer, async, and systems interface. All DOS 1 and 2 functions are in assembler for speed. All video capabilities of PC supported. All printer functions. 65 string functions. Extensive time and date. Directory searches. Polled mode async. (If you want interrupt driven, ask us about the **Greenleaf Comm Library**.) Function key support. Diagnostics. Rainbow Color Text series. Much, much more. **The Greenleaf Functions**. Simply the finest C library (and the most extensive). All ready for you.

THE GREENLEAF FUNCTIONS™


The Library of C Functions that probably has just what you need . . . **TODAY!**

- already has what you're working to re-invent
- already has over 200 functions for the IBM PC, XT, AT, and compatibles
- already complete . . . already tested . . . on the shelf
- already has demo programs and source code
- already compatible with all popular compilers
- already supports all memory models, DOS 1.1, 2.0, 2.1
- already optimized (parts in assembler) for speed and density
- already in use by thousands of customers worldwide
- already available from stock (your dealer probably has it)
- It's called the **GREENLEAF FUNCTIONS**.

The Library of C Functions is Waiting for You

Specify compiler when ordering. Add \$7.00 for UPS second-day air (or \$5.00 for ground). Texas residents add sales tax. Mastercard, VISA, check or P.O. In stock, shipped same day.

<ul style="list-style-type: none"> ■ General Libraries \$185 ■ Comm Library \$185 ■ C/C86 Compiler \$349 ■ Lattice C \$395 ■ Mark Williams \$475 	<p>For Information: 214-446-8641</p> <p>Prices are subject to change without notice.</p>
---	--



GREENLEAF SOFTWARE®

2101 HICKORY DR.
CARROLLTON, TX 75006

Circle no. 43 on reader service card.

than mumbo jumbo going for them. True, they'd made public access to the machines difficult, or allowed it to be difficult. But now that we have our own machines, we find out that there were good reasons for that. It turns out that quite possibly the most difficult thing you can do with a computer is to make it be truly helpful. If you are in a hurry for an answer, you will find it a lot easier to program the machine to take its input in rigid, unforgiving formats, and to give cryptic, insulting error messages, than to

make it flexible and friendly and forgiving.

In fact, we find that in putting on its happy face, the machine uses up so much capacity that it has precious little left in which to do any useful work.

Partly this is a real effect. It truly is difficult to make a computer be helpful, and there are deep, unsolved problems in the way of making that happen. But in part, I think that the revolutionaries were a bit hasty in exiling those priests. They have a genuine science, computer science, that

deals with the elegant and efficient application of machine power. All too little of that science is being applied today. The result is that we have machines with 512K of storage that can't keep up with the demands of a single user. Damn it, we used to support twenty APL users on a single 360 model 40 with 128K; we considered a 370/158 with 512K a big machine, suitable for use by a middle-sized company with half a dozen programmers online to TSO or VM/370. The *big* macro assembler for the IBM 370 needs a 128K partition to run in. That's the assembler program plus all the tables and work areas and buffers it uses. The macro assembler for the IBM PC can barely load *itself* into 128K, and it can't assemble anything of useful size until you give it 256K. In short, there has been a major loss of efficiency in moving from the software that the priests built for the corporate mainframes to the software that we are putting on personal computers today.

As a good revolutionary, I must believe that computer *use* is for everybody, and I really do. But programming the computers so they can *be used* is almost certainly a job for specialists.

The question that confronts us now is: whose specialists are they going to be? Will they be the tools of the new computing establishment, working for the fiscal advantage of a company like IBM or AT&T or Microsoft, or for the political advantage of a nation like Japan or France?

Or will they be grass-roots benefactors like so many who supplied the software we have now—and who often supplied it through the pages of *DDJ*? I would like it to be the latter. In fact, I propose to you that what *DDJ* has always done is to support and train and encourage our grass-roots systems programmers. And that is still the best, and most truly radical, thing that it can do in the future. I'm delighted to see that its editors agree with me. Proof that they do is in their publishing Richard Stallman's "GNU Manifesto" (*DDJ* #101, March 1985), an outrageous, revolutionary challenge quite in the old vein.

My own best writing for *DDJ* has



THE STRUCTURED PROGRAMMING TOOL FOR MODERN TIMES

- Design your programs right on the screen, using modern techniques based on the popular Jackson Structured Programming method (JSP)!
- **DEZIGN** is more than just another flowcharting tool. It is an integrated tool for designing and documenting programs and for generating ADA, C, PASCAL, and PL/I source code, as well as dBASE II and dBASE III command files.
- **DEZIGN** enables you to create Data and Program Structure Diagrams using the Sequence, Selection (IF-THEN-ELSE), and Iteration (DO WHILE) constructs; assign detailed statements to the diagrams; and synthesize source code from the control logic represented on the diagrams and the detailed statements assigned to them.
- **DEZIGN** lists for \$200. It runs on the IBM PC, XT, or AT and requires 128K RAM, one disk drive, and an 80-column color or monochrome display.
 - DEZIGN-PC runs under DOS 2.0, 2.1, and 3.0.
 - DEZIGN-86 runs under CP/M-86 1.1.
- Want to learn more? Please contact us concerning pricing and availability of JSP reference texts and seminars.

ZEDUCOMP • P.O. BOX 68 • STIRLING, NJ 07980
(201) 755-2262

dBASE II and dBASE III are trademarks of Ashton-Tate, Inc.

Circle no. 126 on reader service card.



The Most Powerful C

for the IBM AT • MACINTOSH • MS DOS • CP/M-80 • ROM APPLICATIONS
IBM PC/XT • APPLE II • CP/M-86 • TRSDOS • CROSS DEVELOPMENT

Why Professionals Choose Aztec C

AZTEC C compilers generate fast, compact code. AZTEC C is a sophisticated development system with assemblers, debuggers, linkers, editors, utilities and extensive run time libraries. AZTEC C is documented in detail. AZTEC C is the most accurate and portable implementation of C for microcomputers. AZTEC C supports specialized professional needs such as cross development and ROM code development. MANX provides qualified technical support.

AZTEC C86/PRO

— for the IBM AT and PC/XT

AZTEC C86/PRO provides the power, portability, and professional features you need to develop sophisticated software for PC DOS, MS DOS AND CP/M-86 based microsystems. The system also supports the generation of ROM based software for 8088/8086, 80186, and 80286 processors. Options exist to cross develop ROM code for 65xx, 8080, 8085, and Z80 processors. Cross development systems are also available that target most micro computers. Call for information on AZTEC C86/PRO support for XENIX and TOPVIEW.

POWERFUL — AZTEC C86/PRO 3.2 outperforms Lattice 2.1 on the DHRYSTONE benchmark 2 to 1 for speed (17.8 secs vs 37.1) while using 65% less memory (5.8k vs 14k). The AZTEC C86/PRO system also compiles in 10% to 60% less time and supports fast, high volume I/O.

PORTABLE — MANX Software Systems provides real portability with a family of compatible AZTEC C software development systems for PC DOS, MS DOS, CP/M-86, Macintosh, CP/M-80, APPLE II+, IIe, and IIc (NIBBLE - 4 apple rating), TRSDOS (80-MICRO - 5 star rating), and Commodore C64 (the C64 system is only available as a cross compiler - call for details). AZTEC C86/PRO is compatible with UNIX and XENIX.

PROFESSIONAL — For professional features AZTEC C86/PRO is unparalleled.

- Full C Compiler (8088/8086 - 80186 - 80286)
- Macro Assembler for 8088/8086/80186/80286
- Linkage Editor with ROM support and overlays
- Run Time Libraries - object libraries + source
DOS 1.x; DOS 2.x; DOS 3.x; screen I/O; Graphics;
UNIX I/O; STRING; simulated float; 8087 support;
MATH; ROM; CP/M-86
- Selection of 8088/8086, 80186, or 80286 code generation to guarantee best choice for performance and compatibility

- Utility to convert AZTEC object code or libraries to Microsoft format. (Assembly + conversion takes less than half the time as Microsoft's MASM to produce MS object)
- Large memory models and sophisticated memory management
- Support products for graphics, DB, Screen, & ...
- ROMable code + ROM support + separate code and data + INTEL Hex Converter
- Symbolic Debugger & Other Utilities
- Full Screen Editor (like Vi)
- CROSS Compilers are available to APPLE II, Macintosh, CP/M-80, TRSDOS, COMMODORE C64, and ROM based 65xx, and 8080/8085/Z80
- Detailed Documentation

AZTEC C86/PRO-AT\$500
(configured for IBM AT - options for 8088/8086)

AZTEC C86/PRO-PC/XT\$500
(configured for IBM PC/XT - options for 80186/80286)

AZTEC C86/BAS includes C compiler (small model only), 8086 MACRO assembler, overlay linker, UNIX, MATH, SCREEN, and GRAPHICS libraries, debugger, and editor.

AZTEC C86/BAS\$199
AZTEC C86/BAS (CP/M-86)\$199
AZTEC C86/BAS (DOS + CP/M-86)\$299
UPGRADE to AZTEC C86/PRO\$310
C-TREE Database with source\$399
C-TREE Database (object)\$149

CROSS COMPILERS

Cross Compilers for ROM, MS DOS, PC DOS, or CP/M-86 applications.

VAX -> 8086/80xxx cross\$5000
PDP-11 -> 8086/80xxx cross\$2000

Cross Compilers with PC DOS or CP/M-86 hosts are \$750 for the first target and \$500 for each additional target. Targets: 65xx; CP/M-80; C64; 8080/8085/Z80; Macintosh; TRSDOS; 8086/8088/80186/80286; APPLE II.

AZTEC C68K

— for the Macintosh

For power, portability, and professional features AZTEC C68K-c is the finest C software development system available for the Macintosh.

The AZTEC C68K-c system includes a 68000 macro assembler, a linkage editor, a source editor, a mouse based editor, a SHELL development environment, a library of UNIX I/O and utility routines, full access and support of the Macintosh TOOLBOX routines, debugging aides, utilities, make, diff, grep, TTY simulator with upload & download (source supplied), a RAM disk (for 512K Mac), a resource maker, and a no royalty license agreement. Programming examples are included. (Over 600 pages of documentation).

AZTEC C68K-c requires a 128K Macintosh, and two disk drives (frugal developers can make do with one drive). AZTEC C68K supports the 512K Macintosh and hard disks.

AZTEC C68K-c (commercial system)\$500
AZTEC C68K-p (personal system)\$199
AZTEC C68K-p to AZTEC C68K-c upgrade\$310

Mac C-tree database\$149
Mac C-tree database with source\$399
Lisa Kit (Pascal to AZTEC C68k object converter) ..\$ 99

AZTEC C65

— for the APPLE II

"...The AZTEC C-system is one of the finest software packages I have seen..." NIBBLE review, July 1984.

The only commercial C development system available that runs native on the APPLE II+, IIc, and IIe, the AZTEC C65 development system includes a full floating point C compiler compatible with UNIX C and other MANX AZTEC C compilers, a 6502 relocating assembler, a linkage editor, a library utility, a SHELL development environment, a full screen editor, UNIX I/O and utility subroutines, simple graphics, and screen functions.

AZTEC C65 (Apple DOS 3.3)\$199
AZTEC C65/PRO (Apple DOS + ProDos)\$350
(call for availability)

AZTEC C II/PRO

— for CP/M-80

The first member of the AZTEC C family was the CP/M-80 AZTEC C compiler. It is "the standard" compiler for development on CP/M-80. The system includes the AZTEC C II C compiler, an 8080 assembler, a linkage editor, an object librarian, a full library of UNIX I/O and utility routines, CP/M-80 run time routines, the SMALL library (creates modules less than 3K in size), the fast linker for reduced development times, the ROM library, RMAC and M80 support, library source, support for DRI's SID/ZSID symbolic debugger, and more.

AZTEC C II/PRO\$349
AZTEC CII/BAS\$199
C-TREE Database with source\$399
C-TREE Database in AZTEC object form\$149

AZTEC C80

— for TRSDOS (Radio Shack Model III & 4)

"I've had a lot of experience with different C compilers, but the Aztec C80 Compiler and Professional Development System is the best I've seen." 80-Micro, December, 1984, John B. Harrell III

This system has most of the features of AZTEC C II for CP/M. It is perhaps the best software development system for the Radio Shack Model III and IV.

AZTEC C80 model 3 (no floating point)\$149
AZTEC C80 model 4 (full)\$199
AZTEC C80/PRO (full for model 3 and 4)\$299

To order or for information call:

800-221-0440

(201) 530-7997 (NJ and outside U.S.A.). Or write: MANX SOFTWARE SYSTEMS, P.O. Box 55, Shrewsbury, N.J. 07701.

MANX



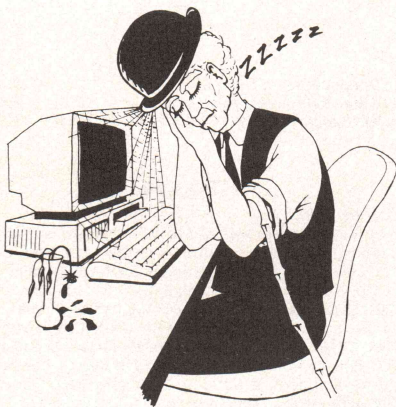
TRS 80 RADIO SHACK TRS DOS is a trademark of TANDY.
APPLE DOS MACINTOSH is a trademark of APPLE.

For Technical Support
(Bug Busters) call: 201-530-6557

SHIPPING INFORMATION - Standard U.S. shipment is UPS ground (no fee). In the U.S. one day shipment is \$20, two days is \$10. Canadian shipment is \$10. Two days shipment outside the U.S. is by courier and is freight collect.

GROWING OLD?

...waiting
for C programs to
compile and link?



Use C-terp the complete C interpreter

**This is the product you've been
waiting (and waiting) for!**

Increase your productivity and avoid agonizing waits. Get instant feedback of your C programs for debugging and rapid prototyping. Then use your compiler for what it does best...compiling efficient code ...slowly.

C-terp Features

- Full K&R C (no compromises)
- Complete built-in screen editor--no half-way house, this editor has everything you need such as multi-files, inter-file move and copy, global searching, auto-indent, tab control, and much more.
- Fast--Linking and semi-compilation are breath-takingly fast. (From edit to run completion in a fraction of a second for small programs.)
- Convenient--Compiling and running are only a key-stroke or two away. Errors direct you back to the editor with the cursor set to the trouble spot.
- Compiler Compatible--You can access functions and externals compiled with C86 or Lattice C or assembly language. Utilize your existing libraries unchanged!
- Complete Multiple Module Support--Instant global searches, auto-compile everything that's changed, etc.
- Many more features including batch mode and symbolic debugging.
- Runs on IBM PC, DOS 2.x, 192K and up
- **Price: \$300.00 (Demo \$45.00) MC, VISA**

Price of demo includes documentation and shipping within U.S. PA residents add 6% sales tax. Specify C86 or Lattice version.

GIMPEL SOFTWARE

3207 Hogarth Lane • Collegeville, PA 19426
(215) 584-4261

*Trademarks: C86 (Computer Innovations), Lattice (Lattice Inc.), IBM (IBM Corp.), C-terp (Gimpel Software)

been inspired by a species of the Robin Hood ethic: a desire to steal ideas from the well-funded and put them into the heads of the unfunded. I enjoyed writing my recent article on Prolog for just that reason. Prolog and the ideas behind it had been the property of a few academics, and these ideas were rapidly being co-opted by the Japanese for their Fifth Generation. But *DDJ* gave me a chance to snatch those same notions and spread them around everywhere, making them accessible to everybody who can grasp them. Of course the real revolutionaries are the people in London who publish *Micro-PROLOG* and the ones in this country who are selling a version of DEC-10 Prolog for the IBM PC. The ideas wouldn't be much use without the cheap software that implements them, and that wouldn't be of any use without the cheap micros to run it on.

So these processes all feed back on themselves and get richer and deeper. But the continuation of this revolution, at least its continuation as a revolution and not as a captive toy of the corporations and the advertisers, depends on getting ever more sophisticated ideas and tools into the hands of creative people, so that they can keep putting ever more sophisticated programs back into ours.

Notice how the ante goes up on every round. The level of sophistication is escalated every time through the loop. The Mac's internal toolkit, and its competitive followers Windows and GEM, are vastly harder to use well than the old CP/M BDOS functions. Graphics, AI, and speech recognition, in their different ways, require the exercise of all the algorithmic tools of computer science. And it takes very abstruse, very mathematical tools to analyze these massive, sophisticated programs and prevent them from overflowing the capacities of personal hardware, which really are quite limited. The experience of mainframers has been that your hardware is *always* an order of magnitude too small for the software you'd like to run. There's no indication that this law has been repealed for micros. We will be squeezing quarts into pint bottles forever,

and it takes sound theoretical insight to do that well.

I should add that the rule that the capacity of the hardware always falls short of the needs of the software just *might* be repealed by highly parallel, multiple-micro systems. Two things about such systems: first, they are right on the fringe of computer science and the academics haven't the foggiest idea how to build good software for them; and second, they are ideally suited for hobbyist-level, garage experiments.

But this is why the face and contents of *DDJ* have been changing. Whether the editors and contributors knew it or not, they've been trying to prepare themselves to discuss and understand computers at the more sophisticated level permitted by the second generation of micros. But there's been no change in who is doing the discussing, nor in their essentially subversive motive, which is to snatch the tools of the establishment and apply them in the public domain.

It's that mischievous, subversive motive that has kept me writing for *DDJ*. That, and the belief I still hold that, somewhere out there, fingers on the keyboard of an Apple or Osborne, is a fifteen-year-old kid who is capable of writing the program that will turn the world upside down. The proper role of *DDJ* is to make sure that that kid has free, unfettered access to the ideas and the software tools that she needs in order to write it.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 191.

The first complete programming environment brings the industry to an all-time low.

\$80.88

Modula-2 has been hailed as the programming language of the future. Its modular design and built-in error-control features make programming more efficient than ever.

And now there's a system that makes programming more affordable than ever. Interface Technologies' Modula-2 Software Development System (M2SDS).

EASY TO LEARN.

EFFICIENT TO USE.

M2SDS features a "syntax-directed" editor that makes programming easy for beginners to learn. And faster for professionals to use.

With our editor you can enter full statements with a single keystroke. And save up to 90% on typing time. It also gives on-line help in correcting undefined variables and syntax errors—which saves even more time.

Multiple editor windows let you refer to one file

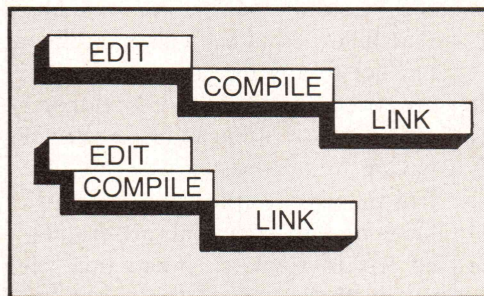
Work faster and easier with multiple editor windows.



while you edit another. That's one more way M2SDS adds hours of more creative, more productive time to your day.

TURN "WAIT TIME" INTO "WORK TIME." When there's no time like real-time, you can count on the M2SDS compiler. Up to 100 lines of Modula-2 text can be turned into native machine code in less than five seconds.

To create programs using your computer's full capacity, there are 18 library modules. And unlike



It not only has a faster compiler; it also saves time by compiling while you edit.

So whether you're a professional looking for a faster way to program, or a novice looking for an easier way to learn, there's a Modula-2 Software Development System just for you.

Call us today for more information or to order your M2SDS. Find out how our new low in system pricing can put your programming efficiency at an all-time high.

WE ACCEPT CHECKS, MASTERCARD, VISA AND AMERICAN EXPRESS. Price does not include shipping and handling. Texas residents add 6.125% Sales Tax. International orders add \$30.

**INTERFACE
TECHNOLOGIES**

3336 Richmond, Suite 200, Houston, TX 77098

other low-priced compilers, M2SDS has a linker that assembles the components of your program. Automatically.

BREAKTHROUGH TECHNOLOGY.

BREAKTHROUGH PRICE. M2SDS works with IBM® PC, XT, AT or any other 100% compatible computer. Any programs you develop, you own. And M2SDS is non-copy protected.

For just \$80.88, M2SDS is the complete programming environment. Including editor, compiler, linker, library modules, 8087 support and more.

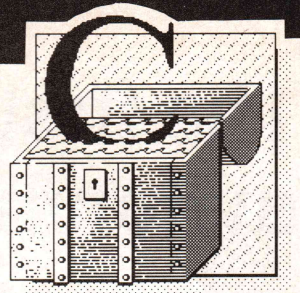
Or choose the expanded, fully upgradeable SDS-XP for just \$249. Later you can add a debugger, foreign object import module and tool box for even more programming capability. And efficiency.

GET MORE PROGRAMMING EFFICIENCY IN A SYSTEM THAT COSTS LESS. IN TEXAS, CALL (713) 523-8422.

CALL 1-800-922-9049

Circle no. 50 on reader service card.

by Allen Holub



A command line switch is an argument usually preceded by a '-' that modifies the way in which a program works. Just about every C program I write uses command line switches, and all these programs have to process these switches one way or another. For a long time, I wrote a different little subroutine for each program, modifying the routine to deal with the idiosyncrasies of a particular program. After writing the same subroutine about a million times (and making the same off-by-one error with `argc` every time), I thought that there *had* to be a better way. All my processing routines were essentially the same in structure; only the names of the command line switches were different. A general purpose command line processor was clearly what I needed.

Looking around in the literature for something that did the job, I found the "Argum" package, which appeared in Anthony Skjellum's C/Unix column in *DDJ* (#70, August 1982). The program was well written and did what I needed, but it had several problems. First of all, it did a lot more than I needed and the extra functions added extra code. Because this package was going to be included in every program I wrote, a reduction in scope seemed to be a good idea. Another, related, problem was the internal tables used by Argum. They were created at run-time (as compared to compile time). This added both extra code and extra execution time to any program that used Argum. Finally, Argum had no convenient way to deal with errors on the command line. In view of these problems, I decided to write my own processor, a description of which follows. This is by far the most frequently used subroutine in my standard library.

Getargs()

The command line processor is actually a package of subroutines. Access to these routines is through a single procedure, called `getargs()`. The routines are table driven. In every program you have to declare a table in which the various command line switches are described. `Getargs()` removes the switches from the command line as it works. This means that position-sensitive arguments (e.g., a switch that applies to all files that follow it on the command line, but not to any files that precede it) are not supported. I don't use this sort of switch very often, and I got tired of having to skip past arguments that had already been processed to get to a filename. The arguments are evaluated left to right, so interaction between PROC type switches (see below) is possible.

When `getargs()` finds an error on the command line, it prints to `stderr` a list of all legal switches, along with a brief description of what those switches do and their default values. After it prints the error message, `getargs()` terminates the program with an `exit(1)` call. This closes any open files and returns to the operating system.

Command Line Switch Formats

Command line switches all must take the form:

```
- <character> [ <number> !
               <string> ]
```

That is, all arguments start with a minus sign; each switch is identified by a single letter that follows the minus sign immediately (no intervening spaces); the letter may be followed by an optional number or string, again with no spaces between the character that serves as the switch identifier

and the corresponding number or string. Switches may be combined, provided that the argument types allow this combination. For example, the command line:

```
program -a -b123 -c
```

can also be written as:

```
program -ab123c
```

However, if `getargs()` expects a string to follow the switch identifier character, then it assumes that the rest of the argument is part of the string. You have to be careful when combining string type command line switches with other types.

Using Getargs()

To use `getargs()` you must do two things: (1) set up a table to tell the routine what kind of command line switches to expect; (2) call the routine itself somewhere early in the `main()` module. The file `getargs.h` (Listing One, page 32) contains the `#defines` and `typedefs` needed to create the command switch descriptor table. This table is an array of structures:

```
typedef struct
{
    unsigned    arg : 7    ;
    unsigned    type : 4   ;
    int         *variable ;
    char        *errmsg    ;
}
ARG;
```

The `arg` field is a single character that identifies the switch on the command line. In the following descriptions this character is represented as `<switch>`. The `type` field may take any one of five values, all of which are

#defined in getargs.h. The behavior of getargs() will vary according to the value.

INTEGER switches take the form

— <switch> <number>

Numbers preceded by 0x are hex, by 0 without the x, octal. All others are decimal. The number is terminated by any character not legal in the indicated radix. Any characters that follow are assumed to be additional switches. The int sized variable pointed to by the variable field of the ARG structure will be set to the value of the <number>. If the <number> isn't typed on the command line, *variable will be set to 0.

BOOLEAN switches will cause some action based on the presence or absence of the indicated switch on the command line. If the switch is present, then the int pointed to by the variable field is set to 1, otherwise *variable is not modified.

CHARACTER switches take the form

— <switch> <character>

When the switch is found on the command line, then *variable is set to the character immediately following the <switch> character.

STRING switches take the form

— <switch> <string>

When the switch is found on the command line, the *character pointer* pointed to by the variable field is set to point at a string consisting of all characters following (but not including) the <switch> character up to the end of the current argument (not to the end of the command line). In the case of combined switches in a single argument, the STRING argument must be the last one because all following characters will be considered part of the <string>. When defining a STRING switch in the table, be sure to cast the variable into an integer pointer. See Listing Three (page 38), line 15, for an example.

PROC switches take the form

— <switch> <anything>

This works like the STRING switch in that all characters following <switch> up to the end of the current argument will be part of the <anything>. However, the variable field is a pointer to a subroutine that is called indirectly as soon as the switch is encountered on the command line. A pointer to <anything> is passed to this subroutine as a single argument. An example of such a subroutine is given in Listing Three, line 19. It is the responsibility of the called subroutine to parse <any-

thing> as appropriate.

The errmsg field is used to print an error message if an undefined switch is found. An example of the format is shown in the Figure (page 30). This message was generated when the command line "argtest -x" was given to argtest.

Listing Three (if you haven't realized it by now) is an example of how to use getargs(). Lines 5-9 are declarations of the objects that are used in the variable fields of the switch descriptor table. The initial values of

You're in Good Company When You Program in BetterBASIC



All of these companies rely on BetterBASIC to write their software programs. They have found that BetterBASIC combines the features they need from BASIC, Pascal, C and Fortran in one familiar environment. Some of these features include the following.

640K Now you can use the full memory of your PC to develop large programs.

STRUCTURED Create well organized programs using procedures and functions that are easily identified and understood and completely reusable in future programs.

MODULAR Use procedures and functions grouped together to form "library modules."

INTERACTIVE BetterBASIC acts like an interpreter, responding to the users' commands in an immediate mode. However, each statement is actually compiled as it is entered.

EXTENSIBLE Create your own BetterBASIC modules which contain BetterBASIC extensions (ideal for OEMs).

COMPILED Each line of the program is compiled as it is entered

into the computer's memory rather than interpreted at runtime. The optional Runtime System generates EXE files.

BetterBASIC Runs on IBM PC, IBM PC/XT and compatibles.

CALL 1-800-225-5800 Order Better BASIC now, or write Summit Software Technology, Inc.™, P.O. Box 99, Babson Park, Wellesley, MA 02157. Prices are listed below.

BetterBASIC: \$199 Runtime System: \$250
8087 Math Module: \$99

Still not convinced? Order the BetterBASIC sample disk which includes a demo, a tutorial, compatibility issues, 50 lines of BetterBASIC and more. Only \$10.

MasterCard, VISA, P.O. Checks, Money Order, C.O.D. accepted.

BetterBASIC is a registered trademark of Summit Software Technology, Inc.

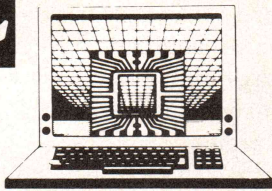
IBM PC and IBM PC/XT are registered trademarks of International Business Machines Corp. Tandy is a registered trademark of Tandy Corp. Illustrated above are registered trademarks of the following companies: Mobil Oil Corp.; A T & T; General Electric Co.; Westinghouse Electric Corp.; TRW, Inc.

Better BASIC™

ALSO AVAILABLE FOR THE TANDY 2000, 1200 AND 1000

Circle no. 98 on reader service card.

C PROGRAMMERS



db_VISTA

The first DBMS designed exclusively for the C language.

PREFERRED over ISAM and file utilities
POWER like a mainframe DBMS
PRICED like a microcomputer utility
PORTABILITY like only C provides

FEATURES INCLUDE:

- Written in C, for C.
- Maximum data efficiency using the network database model.
- Virtual memory disk accessing.
- Fast B*-tree indexing method.
- Multiple key records-any or all data fields may be keys.
- ROYALTY FREE RUNTIME.
- SOURCE CODE INCLUDED.
- Three month extended applications support included.

FREE OFFER

MENTION THIS AD and choose any one of the following C tools from Lattice at no additional charge, when you order db_VISTA

- Lattice C Compiler
- C-Sprite Program Debugger
- Lattice Window Manager
- Curses Unix-compatible Screen Manager (source code included)
- Panel Forms Manager
- CVUE Screen Editor

db_VISTA with source code: \$495
db_VISTA without source code: \$395

OR

COMPLETE C Development package including:

db_VISTA, Lattice C compiler, C-Sprite, CVUE, & Curses

a \$1520.00 value for only \$895.00

db_VISTA available for PC-DOS/MS-DOS, for most popular C compilers including Lattice, DeSmet, Computer Innovations, AZTEC. Also available for most Unix systems and CTOS.

RAIMA

CORPORATION
11717 Rainier Avenue South
Seattle, WA 98178
206/772-1515

CALL TOLL-FREE
1-800-843-3313
at the tone: 700-992
ask for Jim

MONEY BACK GUARANTEE

Circle no. 83 on reader service card.

these variables remain unchanged if the switch is not encountered on the command line at runtime. The table itself, Argtab, is declared on lines 10-17. The main() routine calls getargs() on line 35. The rest of main() just prints the command line, both before and after the getargs() call, so you can see how getargs() strips processed switches out of argv.

Getargs() itself starts on line 98 of Listing Two (page 32). It is passed four arguments: argv, argc, a pointer to the switch descriptor table (tabp), and the size of this table in elements (not bytes). The routine returns a new value of argc, and argv is compressed, i.e., all entries containing command line switches are removed from it. The for loop on line 112 processes argv one element at a time. Nargv points into argv and strips processed switches. Nargv is initialized to point at argv[1] because argv[0] can't possibly contain a command line switch (it holds the program name). If no leading minus sign is found in the argument, *argv is copied to *nargv and both pointers are advanced. In addition, nargc (new argc) is incremented. If the argument does begin with a minus sign, it is processed as a command line switch. In this case *argv is not copied to *nargv, and argv, but not nargv, is advanced. This effectively eliminates the argument containing the switch from the argv array.

Findarg() (Listing Two, line 44) is used by getargs() to see if an argument is in the table. It performs a linear search. Because the table is usually fairly small, it seemed as if the extra code needed for a binary search

wasn't justified. Findarg() takes three arguments: c is the switch identifier character being searched for; tabp and tabsize are the same parameters as were passed to getargs().

Setarg() (Listing Two, line 9) is called when a command line switch is found in the table. Argp is a pointer into the table, as returned from findarg(). Linep is a pointer into the argv entry that is being processed. Casts were used to make this routine as transportable as possible.

Pr_usage() (Listing Two, line 57) is used to print the error message shown in the Figure when an illegal command line switch is found. It just spins through the table, printing the current contents of the object pointed to by the variable field as well as the message given in the errmsg field.

The final routine in the package is stoi() (for string to integer, Listing Four, page 38), which is used to process INTEGER switches. Stoi() is a fancy version of the standard library routine atoi() (described on page 58 of Kernighan & Ritchie). Stoi() can handle numbers in base 8, 10, or 16. If code size is a real issue, you may want to remove the code that processes octal numbers (Listing Four, lines 45-53). Another major difference between stoi() and atoi() is that stoi() is passed a *pointer* to a character pointer. That is, it is passed the *address* of the pointer, which in turn points into the string to be processed. This extra level of indirection lets you modify the character pointer itself to point past the end of the number being processed. If you use atoi(), you have to go through the string twice, once to extract the number and once

Illegal argument <x>. Legal arguments are:

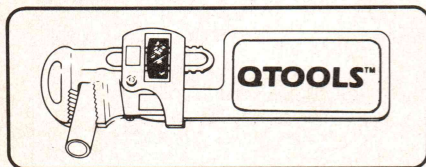
-b	boolean argument	(value is FALSE)
-c<c>	character argument	(value is .)
-n<num>	integer argument	(value is 0)
-s<str>	string argument	(value is <doo wha>)
-p<str>	procedure argument	

Figure
Error Output from Program in Listing Three

Let Us Mind Your Q's and T's

Quality Tools from QCAD Systems, Inc.

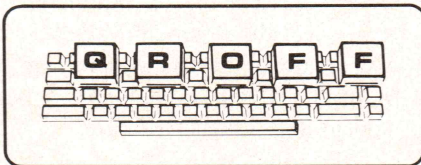
Tools



A UNIX-like toolset that runs under MS-DOS, making full use of DOS I/O redirection and environment variables. Great for pattern search, substitution, translation, file listings, and maintenance. On-line help and detailed user manual with extensive examples. Increases programmer productivity by providing a concise way to specify complex file manipulations. Some tools are versions of UNIX programs -- cat, diff, echo, fgrep, grep, head, ls, more, mv, pr, split, subst, tail, wc -- while others are new; 19 in all! *Runs on the IBM PC.*

\$49.50

Typesetting



A typesetting system for the IBM PC and HP LaserJet printer. Ordinary text files, with some simple command strings added, can be printed quickly and easily with any of these features: mixed fonts, underlining, headers, footers, superscripts, subscripts, left & right margin alignment, page numbers, line spacing, indentation, automatic index generation, table formatting, spacing up or down, and more! Documents, letters, memos, and tables can be formatted and printed as nicely as a print shop could do them! (Requires HP font cartridge -- call for details.) *Runs on the IBM PC and HP 9816.*

\$79.95

Translation



An LALR(1) parser generator that works from grammar, semantics, and skeleton files. Skeleton files provided for C and Pascal; easily adaptable to other languages. Multi-level interactive debugging support. Advanced error recovery mechanism. Sample translators included. Generates a complete parser and lexical analyzer in source code. Extensive documentation and examples. Turbo Pascal included. Educational discount available. Accompanying textbook (2nd ed.) available in late 1985. *Runs on the IBM PC, DEC Rainbow, and HP 9816.*

Full System \$995.00
Demo System \$ 25.00

WRITE OR CALL FOR MORE INFORMATION.

TOLL FREE LINE: 800-538-9787

In California, call 408-255-5574. California residents please add sales tax. Check, money order, Mastercard, and Visa accepted. [UNIX is a trademark of AT & T Bell Laboratories. IBM PC is a trademark of IBM Corp. Turbo Pascal is a trademark of Borland International. LaserJet is a trademark of Hewlett-Packard Co. QTOOLS, QROFF, and QPARSER are trademarks of QCAD Systems, Inc.]

QCAD
SYSTEMS, INC.

1164 Hyde Ave., San Jose, CA 95129

Circle no. 51 on reader service card.

more to skip past the characters that represent that number.

Conclusion

Several additions could be made to `getargs()`, at the cost of an increase in complexity and code size.

- A common command line function is to open or close a file in a particular mode and to abort if the file can't be opened. At present I'm doing this with a PROC type command line switch, but you could add another argument type, whose variable field points to a FILE pointer. Alternately, variable could point to a structure that included

the FILE pointer, an open mode, a pointer to an error processing routine, and a default file name.

- Another nice feature would be to mark an argument if its presence is required on the command line. You could accomplish this by adding additional types (i.e., REQ_INTEGER), or by adding another field to the ARG structure. An error message would be printed if the argument wasn't found on the command line.
- `Getargs()` depends on initializers to set default argument values. If your compiler doesn't support initializers, you can add a default value field to the ARG structure.

- I have violated my smallness rule by using `stoi()` to process numeric arguments. If you expect only decimal numbers, you may want to replace `stoi()` with `atoi()`. However, `stoi()` is a useful routine in its own right.

I'm sure that you can think of other bells and whistles. I've been using `getargs()` for a couple years now and am satisfied with it in its existing state. Defining the way the command line looks and then getting switches from it is now a painless process. DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 192.

C Chest (Text begins on page 28)

Listing One

```

1:  /*      Getargs.h      Typedefs and defines needed for getargs
2:  */

3:  #define INTEGER      0
4:  #define BOOLEAN      1
5:  #define CHARACTER    2
6:  #define STRING       3
7:  #define PROC         4

8:  typedef struct
9:  {
10:     unsigned          arg : 7 ;      /* Command line switch      */
11:     unsigned          type : 4 ;     /* variable type            */
12:     int               *variable ;    /* pointer to variable      */
13:     char              *errmsg ;      /* pointer to error message */
14:  }
15:  ARG;
```

End Listing One

Listing Two

```

1:  /*      GETARGS.C      Command line argument processor for C programs
2:  *
3:  *      (C) Copyright 1985, Allen I. Holub. All rights reserved.
4:  *      This program may be copied for personal, non-profit use only.
5:  */

6:  #include <stdio.h>
7:  #include <getargs.h>

8:  typedef int          (*PFI)();

9:  static char          *setarg( argp, linep )
10:  ARG                  *argp;
11:  char                 *linep;
12:  {
13:     /*      Set an argument. argp points at the argument table entry
14:     *      corresponding to *linep. Return linep, updated to point
15:     *      past the argument being set.
16:     */

17:     ++linep;

18:     switch( argp->type )
```

```

19:         {
20:         case INTEGER:
21:             *argp->variable = stoi( &linep );
22:             break;

23:         case BOOLEAN:
24:             *argp->variable = 1;
25:             break;

26:         case CHARACTER:
27:             *argp->variable = *linep++ ;
28:             break;

29:         case STRING:
30:             *(char **)argp->variable = linep ;
31:             linep = "";
32:             break;

33:         case PROC:
34:             (* (PFI)(argp->variable) )( linep );
35:             linep = "";
36:             break;

37:         default:
38:             fprintf(stderr, "INTERNAL ERROR: BAD ARGUMENT TYPE\n");
39:             break;
40:     }
41:     return( linep );
42: }

43: /*-----*/

44: static ARG      *findarg( c, tabp, tabsize )
45: int             c, tabsize;
46: ARG             *tabp;
47: {
48:     /*      Return pointer to argument table entry corresponding
49:     *      to c (or 0 if c isn't in table).
50:     */
51:
52:     for(; --tabsize >= 0 ; tabp++ )
53:         if( tabp->arg == c )
54:             return tabp;

55:     return 0;
56: }

57: static pr_usage( tabp, tabsize )
58: ARG      *tabp;
59: int      tabsize;
60: {
61:     /*      Print the argtab in the form:
62:     *      -<arg> <errmsg>      (value is <*variable>)
63:     */
64:
65:     for(; --tabsize >= 0 ; tabp++ )
66:     {
67:         switch( tabp->type )
68:         {
69:             case INTEGER:
70:                 fprintf(stderr, "-%c<num> %-40s (value is ",
71:                     tabp->arg, tabp->errmsg);
72:                 fprintf(stderr, "%-5d)\n", *(tabp->variable) );
73:                 break;

74:             case BOOLEAN:
75:                 fprintf(stderr, "-%c      %-40s (value is ",
76:                     tabp->arg, tabp->errmsg);
77:                 fprintf(stderr, "%-5s)\n", *(tabp->variable)
78:                     ? "TRUE": "FALSE");
79:                 break;

80:             case CHARACTER:
81:                 fprintf(stderr, "-%c<c>      %-40s (value is ",
82:                     tabp->arg, tabp->errmsg);
83:                 fprintf(stderr, "%-5c)\n", *(tabp->variable) );
84:                 break;

85:             case STRING:
86:                 fprintf(stderr, "-%c<str> %-40s (value is ",
87:                     tabp->arg, tabp->errmsg);

```

(Continued on page 36)

dBASE III v

MORE POWERFUL!

dBASE III IS A ***DRAMATIC AND POWERFUL*** NEW BUSINESS PROGRAMMING LANGUAGE.

—Bob Davies, President/STB Corporation

AFTER A TWO-HOUR HANDS-ON SESSION WITH dBASE III, IT IS ***EVIDENT*** THAT ASHTON-TATE HAS MADE A ***SIGNIFICANT*** ADVANCEMENT IN THE TECHNOLOGY OF MICROPROCESSOR-BASED DBMS.

—Robert Dew, Vice President/The Computer Society

RATHER THAN BEING AN ***IMITATOR***, ASHTON-TATE HAS ONCE AGAIN SHOWN ITSELF TO BE AN ***INNOVATOR***.

—Larry Heimindinger/Origin, Inc.

ASHTON-TATE HAS LISTENED TO THEIR USERS. THIS PRODUCT ***ADDRESSES*** EVERY ITEM ON MY ***WISH-LIST***.

—Mark DeVia/National Microware, Inc.

ASHTON-TATE HAS USHERED IN A ***RENAISSANCE***. THE NEXT ***GENERATION*** OF SOFTWARE IS NOW A ***REALITY***.

—Chris MacNeil/Abel Computers

THE NEW REPORT GENERATOR IS ***SUPER*** TO USE AND MODIFY. LABEL GENERATION IS ALSO A NICE TOUCH.

—Michael Broska/Agate Systems, Inc.

dBASE III™ is the powerful and easy-to-use relational database management system you've been waiting for. You can use it without hesitation whether you're a beginner or an expert.

The big winner in the easy-to-use vs. more powerful controversy is you.

If you want to know all about dBASE III, come to your Ashton-Tate software dealer for a free demonstration. For more information call (800) 437-4329 ext. 2333 or in Colorado call (303) 799-4900 ext. 2333.

dBASE III and Ashton-Tate are trademarks of Ashton-Tate. © Ashton-Tate 1984. All rights reserved.

s. dBASE III

EASY TO USE!

dBASE III IS BY FAR THE *EASIEST*, MOST *COST-EFFECTIVE* WAY TO MANAGE A LARGE DATA BASE.

—Robb Auspitz/McEntyre Designs

dBASE III IS MUCH *EASIER* TO OPERATE AND UNDERSTAND. PROGRAMMING WITH IT IS A *BREEZE*.

—Michael Broska/Agate Systems, Inc.

THE NATURAL CHOICE OF THE NOVICE USER WITH EXPANDABLE NEEDS. dBASE III IS REALLY *EASY TO USE!*

—Alex Gersen/Alex Systems

dBASE III SUCCESSFULLY COMBINES THE FEATURES OF A *POWERFUL* DATABASE MANAGER WITH THE SIMPLICITY AND USER-FRIENDLINESS OF A FILE MANAGER. IT SETS THE STANDARD AGAINST WHICH ALL OTHERS WILL HAVE TO BE *COMPARED*.

—Jerry Schneider, Vice President/WBS & Associates

ASHTON-TATE HAS *OBVIOUSLY* LISTENED TO THE END USERS.

—Dave Browning, Chairman/
Database SIG, Capital PC Users Group

THE DEGREE OF RESPONSIVENESS WHICH ASHTON-TATE HAS SHOWN IN ITS *WILLINGNESS TO LISTEN* AND ADAPT TO USER *NEED* SHOULD GUARANTEE IT A COMMITTED FOLLOWING IN THE MICROCOMPUTER COMMUNITY.

—Phillip Wood, Director of Data Processing/Search Institute

Software from

ASHTON-TATE™

We'll put you in control.

C Chest

(Listing Continued, text begins on page 28)

Listing Two

```
87:             fprintf(stderr, "<%s>)\n",
88:                                     *(char **)tabp->variable);
89:             break;

90:         case PROC:
91:             fprintf(stderr, "-%c<str> %-40s\n",
92:                                     tabp->arg, tabp->errmsg);
93:             break;
94:         }
95:     }
96: }
97: #define ERRMSG "Illegal argument <%c>.  Legal arguments are:\n\n"

98: int      getargs(argc, argv, tabp, tabsize )
99: int      argc, tabsize ;
100: char    **argv ;
101: ARG     *tabp ;
102: {
103:     /* Process command line arguments. Stripping all command line
104:      * switches out of argv. Return a new argc. If an error is found
105:      * exit(1) is called (getargs won't return) and a usage message
106:      * is printed showing all arguments in the table.
107:      */

108:     register int      nargc      ;
109:     register char    **nargv, *p ;
110:     register ARG     *argp      ;

111:     nargc = 1 ;
112:     for(nargv = ++argv ; --argc > 0 ; argv++)
113:     {
114:         if( **argv != '-' )
115:         {
116:             *nargv++ = *argv ;
117:             nargc++;
118:         }
119:         else
120:         {
121:             p = (*argv) + 1 ;

122:             while( *p )
123:             {
124:                 if(argp = findarg(*p, tabp, tabsize))
125:                     p = setarg( argp, p );
126:                 else
127:                 {
128:                     fprintf(stderr, ERRMSG, *p );
129:                     pr_usage( tabp, tabsize );
130:                     exit( 1 );
131:                 }
132:             }
133:         }
134:     }
135:     return nargc ;
136: }
```

End Listing Two

Listing Three

```
1:  /*      ARGTEST.C      Test program for getargs.
2:  */

3:  #include <stdio.h>
4:  #include "getargs.h"

5:  int      boolarg = 0;
6:  int      chararg = '.';
7:  int      intarg = 0;
8:  char    *strarg = "doo wha" ;
9:  extern  proc();

10: ARG      Argtab[ ]=
```

(Continued on page 38)

Add EDITING to your Software with CSE Run-Time™

Your program can include all or a portion of the *C Screen Editor (CSE)*.

CSE includes all of the basics of full screen editing plus source in C for only \$75. For only \$100 more get CSE Run-Time to cover the first 50 copies that you distribute.

Use capabilities like Full cursor control, block move, insert, search/replace or others. Portability is high for OSes, terminals, and source code.

Call for the "CSE Technical Description" and for licensing terms and restrictions. Versions for PC DOS, MSDOS, CPM, more.

Full Refund if
not satisfied in
first 30 days.

Call 800-821-2492

**Solution
Systems**

335-D Washington Street
Norwell, MA 02061
617-659-1571

Circle no. 75 on reader service card.

C Helper™

FIRST-AID FOR C PROGRAMS

Save time and frustration when analyzing
and manipulating C programs. Use C HELPER's
UNIX-like utilities which include:

DIFF and **cmp** – for "intelligent" file comparisons.
XREF – cross references variables by function and line.
C Flow Chart – shows what functions call each other.
C Beautifier – make source more regular and readable.
GREP – search for sophisticated patterns in text.

There are several other utilities that help with converting from one C compiler to another and with printing programs.

C Helper is written in portable C and includes both full source code and executable files for \$135 for MS-DOS, IBM AT CPM-80 or CPM-86. Use VISA, Master Card or COD.

Call: 800-821-2492

**Solution
Systems**

335-D Washington Street
Norwell, MA 02061
617-659-1571

Circle no. 94 on reader service card.

PROFESSIONAL PROGRAMMER'S BULLETIN:

Be Productive, Be

BRIEF™

The Programmer's Editor

TRY BRIEF "RISK-FREE"
FOR 30 DAYS WITH
OUR MONEY-BACK
GUARANTEE!

BRIEF's power and flexibility provide dramatic increases in programming productivity. BRIEF's ergonomically designed human interface becomes a natural extension of your mind, allowing you to eliminate tedium and concentrate on creativity.

- WINDOWS
- Full UNDO (N Times)
- Compile within BRIEF
- Keystroke Macros
- Exit to DOS inside BRIEF
- Programmable Macro Language
- Multiple files, unlimited size
- "Regular Expression" search
- Reconfigure keyboard
- Language sensitive user controllable features (such as Auto-Indent for C)

AVAILABLE FOR PC-DOS, IBM-AT,
AND COMPATIBLE SYSTEMS

ONLY \$195.

DEMO AVAILABLE FOR ONLY \$10
(applicable to future purchase)

CALL TOLL FREE
800-821-2492

for "Technical Description" or to order,

**Solution
Systems**

335-D Washington St., Norwell, MA 02061
617-659-1571

BRIEF is a trademark of UnderWare.
Solution Systems is a trademark of Solution Systems

Circle no. 93 on reader service card.

PROLOG-86™

Become Familiar in One Evening

Thorough tutorials are designed to help learn the PROLOG language quickly. The interactive PROLOG-86 Interpreter gives immediate feedback. In a few hours you will begin to feel comfortable with it. In a few days you are likely to know enough to modify some of the more sophisticated sample programs.

Sample Programs are Included like:

- an EXPERT SYSTEM
- a NATURAL LANGUAGE INTERFACE
(it generates a dBASE II "DISPLAY" command)
- a GAME (it takes less than 1 page of PROLOG-86)

PROTOTYPE Ideas and Applications QUICKLY

1 or 2 pages of PROLOG is often equivalent to 10 or 15 pages in "C" or PASCAL. It is a different way of thinking.

Describe the FACTS and RULES without concern for what the computer will have to do. Maybe you will rewrite in another programming language when you are done.

Programming Experience is not required but a logical mind is. PROLOG-86 supports the de facto STANDARD established in "Programming in Prolog."

AVAILABILITY: PROLOG-86 runs on MSDOS, PC DOS, IBM AT or CPM-86 machines. We provide most formats. The price of PROLOG-86 is **only \$125.**

Full Refund if not
satisfied during
first 30 days.
800-821-2492

**Solution
Systems**

335-D Washington Street
Norwell, MA 02061
617-659-1571

Circle no. 95 on reader service card.

Listing Three

```

11:  {
12:      { 'b', BOOLEAN, &boolarg, "boolean argument" },
13:      { 'c', CHARACTER, &chararg, "character argument" },
14:      { 'i', INTEGER, &intarg, "integer argument" },
15:      { 's', STRING, (int *)&strarg, "string argument" },
16:      { 'p', PROC, (int *)&proc, "procedure argument" }
17:  };

18:  #define TABSIZE ( sizeof(Argtab) / sizeof(ARG) )

19:  proc( str )
20:  char *str;
21:  {
22:      printf("Inside procedure called by -p command line switch, ");
23:      printf("string = <%s>\n", str );
24:  }

25:  main(argc, argv)
26:  int argc;
27:  char **argv;
28:  {
29:      register int i;

30:      printf("Argc == %d. ", argc);
31:      printf("Cmd line: argtest ");
32:      for( i = 1 ; i < argc ; printf("%s ", argv[i++]) )
33:          ;
34:      printf("\n");

35:      argc = getargs( argc, argv, Argtab, TABSIZE) ;

36:      printf("Argc == %d. ", argc);
37:      printf("Cmd line: argtest ");
38:      for( i = 1 ; i < argc ; printf("%s ", argv[i++]) )
39:          ;
40:      printf("\n");
41:  }

```

End Listing Three**Listing Four**

```

1:  /* STOI.C      More powerful version of atoi.
2:  *
3:  *      Copyright (C) 1985 by Allen Holub. All rights reserved.
4:  *      This program may be copied for personal, non-profit use only.
5:  */

6:  #define islower(c)      ( 'a' <= (c) && (c) <= 'z' )
7:  #define toupper(c)      ( islower(c) ? (c) - ('a' - 'A') : (c) )

8:  int      stoi(instr)
9:  register char *instr;
10: {
11:     /*      Convert string to integer. If string starts with 0x it is
12:     *      interpreted as a hex number, else if it starts with a 0 it
13:     *      is octal, else it is decimal. Conversion stops on encounteri
14:     *      the first character which is not a digit in the indicated
15:     *      radix. *instr is updated to point past the end of the number
16:     */

17:     register int      num = 0 ;
18:     register char      *str ;
19:     int      sign = 1 ;

20:     str = *instr;

21:     while(*str == ' ' || *str == '\t' || *str == '\n' )
22:         str++ ;

23:     if( *str == '-' )
24:     {
25:         sign = -1 ;
26:         str++;

```

```

27:     }
28:     if(*str == '0')
29:     {
30:         ++str;
31:         if (*str == 'x' || *str == 'X')
32:         {
33:             str++;
34:             while( ('0' <= *str && *str <= '9') ||
35:                    ('a' <= *str && *str <= 'f') ||
36:                    ('A' <= *str && *str <= 'F') )
37:             {
38:                 num *= 16;
39:                 num += ('0' <= *str && *str <= '9') ?
40:                     *str - '0'
41:                     : toupper(*str) - 'A' + 10 ;
42:                 str++;
43:             }
44:         }
45:         else
46:         {
47:             while( '0' <= *str && *str <= '7' )
48:             {
49:                 num *= 8;
50:                 num += *str++ - '0' ;
51:             }
52:         }
53:     }
54:     else
55:     {
56:         while( '0' <= *str && *str <= '9' )
57:         {
58:             num *= 10;
59:             num += *str++ - '0' ;
60:         }
61:     }
62:     *instr = str;
63:     return( num * sign );
64: }

```

End Listings

2 Megabyte SemiDisk!

Have you been waiting on your slow floppy disk drives too long? SemiDisk Systems has a disk emulator for you! It'll put you in the fast lane, with ultra-fast data transfer, huge storage capacity, convenient battery backup, and a handy print spooler.

Have you been waiting for a SemiDisk big enough to handle your large applications programs, files, and databases - all at once? Your wait is over. SemiDisk Systems is now delivering 2 megabytes of disk storage on a single board!

512k, 1Meg and 2Megabyte SemiDisks are available for S-100 computers, (including the H/Z-100 operating under Z-DOS), IBM PC, XT, & AT, the TRS-80 Models II, 12, & 16, and the Epson QX-10. Once you've tried a SemiDisk you'll know why we say. . .

Someday you'll get a SemiDisk.

Until then, you'll just have to wait.

	<u>512K</u>	<u>1Mbyte</u>	<u>2Mbyte</u>
SemiDisk I, S-100	\$995	\$1795	
SemiDisk II, S-100	\$1295	\$2095	\$2549
IBM PC, XT, AT	\$945	\$1795	\$2499
QX-10,QX-16	\$799		\$2499
TRS-80 II,12,16	\$995	\$1795	\$2499
Battery Backup Unit	\$150		

SEMI Disk

SemiDisk Systems, Inc.

P.O. Box GG, Beaverton, Oregon 97075

503-642-3100



Call 503-646-5510 for CBBS/NW, 503-775-4838 for CBBS/PCS, and 503-649-8327 for CBBS/Aloha, all SemiDisk-equipped computer bulletin boards (300/1200 baud). SemiDisk, SemiSpool trademarks of SemiDisk Systems.

Circle no. 85 on reader service card.

Using Decision Variables in Graphics Primitives

by Tom Hogan

"Graphics algorithms" is not a listing you'll find in the Collected Algorithms of the ACM, but graphics programming presents unique problems to the software developer.

Picture a vast subterranean cavern packed with robed, grey-bearded men. A small platform stands at one end. Billows of purple smoke appear and out of the mist steps a tall man with white hair. He speaks and his voice fills the cavern.

"Welcome, brethren. Tonight we shall consider how to guard our Realme of Graphix from the uninitiated who would learn our Mysteries and divulge them. We will speak of Master Bresenham, who revealed to us the Secrets of ye Mystik Circle."

Does the idea of graphics wizards jealously guarding their secrets seem strange? I can tell you that it didn't seem so to me when I searched the literature for a routine that would map ellipses and couldn't find one.

Light in the Tunnel

Thanks to a tip, I learned about *Fundamentals of Interactive Computer Graphics* by James D. Foley and Andries Van Dam (Addison Wesley, 1982). It contains an algorithm for plotting a circle using a decision variable (based on Bresenham's algorithm). I used the decision variable method (hereafter referred to as the DV method) to derive the algorithm for plotting ellipses that is found in Listing One (page 45).

The DV Method Generalized

Although the DV method was used only to plot an ellipse, it can be used to plot other conic sections as well. In fact, the DV method can be used to generate any well-behaved curve that can be expressed as $G(x, y) = 0$.

The DV Method Compared

Speed is important in graphics applications. The DV method is fast because it adds, shifts, subtracts, and multiplies integers, taking much less time than would be required by floating-point calculations.

An obvious method for plotting an ellipse is to calculate each point independently by incrementing x in unit steps and calculating y using square roots. This method is slow and generates a curve that is nonuniform when the slope of its tangent line falls below -1 (see Figure 1, page 41).

In contrast to the preceding method, the DV method does not calculate points independently. Instead, each succeeding point is mapped with reference to the point previously mapped. Furthermore, this method guarantees that succeeding points are adjacent. Therefore, the generated curve is more uniform than the curve generated by calculating points independently (see Figure 2, page 41).

Reduce the Number of Sample Points

Eight adjacent points surround any previously mapped point. In order to use the DV method to map the curve, you must reduce the number of sample points from eight to two. The sign of the decision variable can only be posi-

Tom Hogan, C Source, 12801 Frost Road, Kansas City, MO 64138.

tive or negative. The two possibilities correspond to the two sample points.

Take the case of an ellipse mapped only in the first quadrant. Points 1, 2, 3, 4, and 6 (see Figure 3, page 42) can be discarded immediately when mapping from Point A to C (see Figure 4, page 42), because x never decreases and y never increases as the ellipse is mapped in that direction. Points 1 through 3 would require y to increase, while points 1, 4, and 6 would require x to decrease.

This reduces the choice to three points: 5, 7, and 8. Which two are chosen as the set of sample points depends on the slopes of the line segments joining them and the last point plotted: the slope of the tangent line must fall within the range of the slopes of the two line segments for the region plotted. The slopes of the line segments of any two points must approach the slope of the tangent line as closely as possible. Thus, a set comprising Points 5 and 7 would be excluded because it would violate this requirement in the region close to B in Figure 4. There remain the possible combinations 5 and 8 and 7 and 8. Points 5 and 8, corresponding to Points S and T in Figure 5 (page 42), can be said to comprise set A. Points 7 and 8, corresponding to Points T and U in Figure 5, can be said to comprise set B.

Changing Sets of Sample Points

From Point A to B (Figure 4), set A is used because the slope of the tangent falls within the range delimited by the slopes of \overline{PS} and \overline{PT} (Figure 5). The decision variable is initialized to its value at Point A (see Equation 9 or 10 in the Table on page 44). The quantities to be added to it are based on the sample points in set A (Equations 6 and 8).

When the slope of the tangent line reaches -1 (at Point B in Figure 4), set B is used (see Figure 6, page 42). The value of the decision variable is recalculated (Equation 11) at Point B for set B. The quantities that are added to the decision variable are now based on set B (Equations 13 and 15).

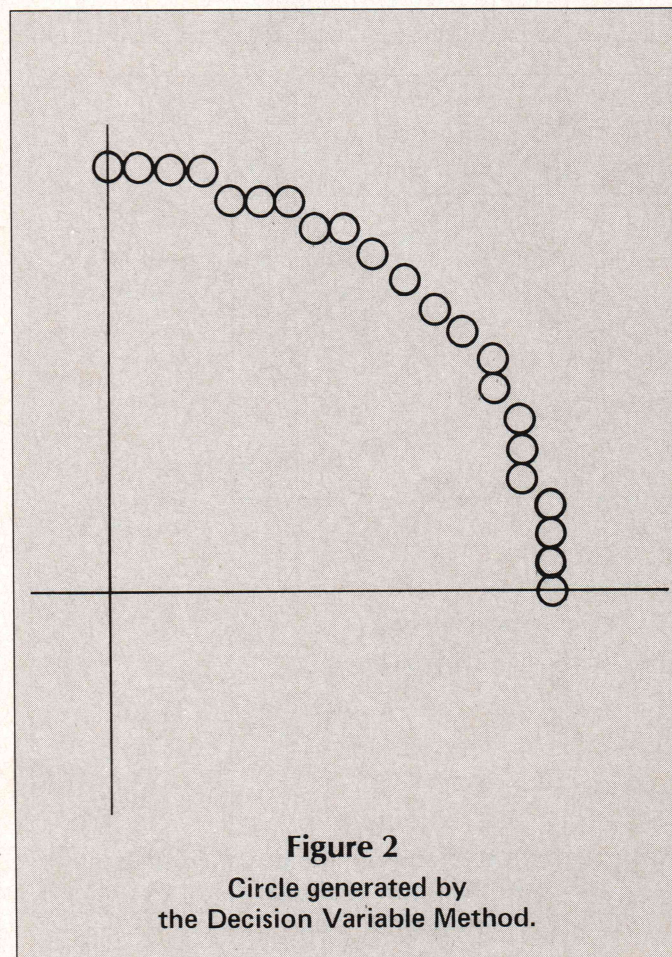
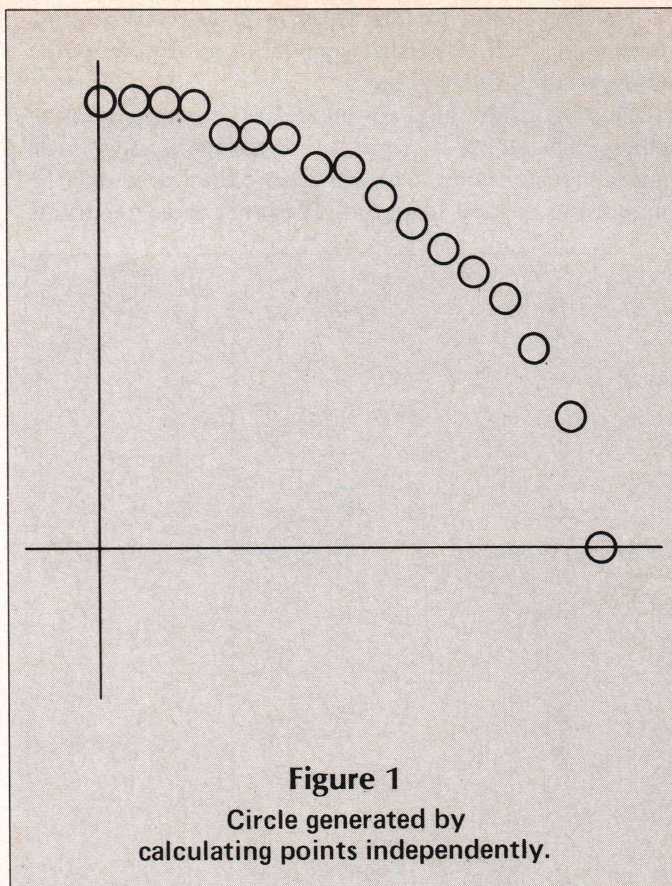
Expansion on General Applications

Please note a key feature of the DV method: the choice of the sample points depends on the slope of the tangent line and the direction of its change. A curve can be broken up into sections for plotting, based on the range of the slope of the tangent line in each section. The method is useful for a broad spectrum of curves.

Example: The Ellipse

The initial value of y depends on the position of the major axis. If it is vertical, y is R_m , half the length of the major axis. If horizontal, y equals R_m multiplied by the aspect ratio. The positions of the major and minor axes depend on the aspect ratio. If it is less than one, then the major axis is horizontal. If greater than one, it is vertical.

Long variables were used in this routine instead of integers or doubles in order to provide the widest range of values for the aspect ratio and R_m while retaining reasonable plotting speed. Integers limited the ellipse's size too much, while doubles were too slow.



The lower limit for the aspect ratio is 0.004 and R_m must not exceed 154. Otherwise, the values of some of the long variables will overflow.

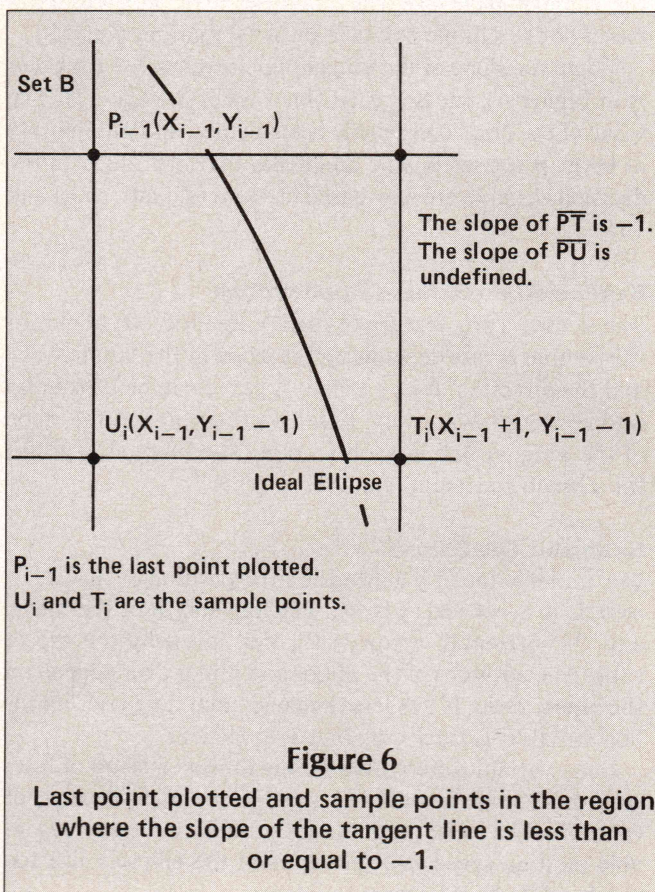
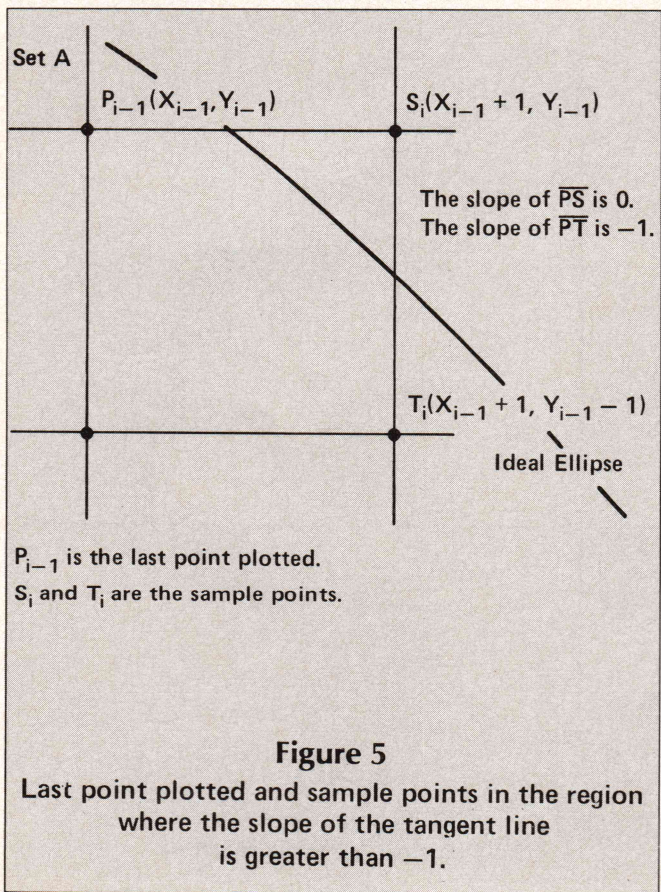
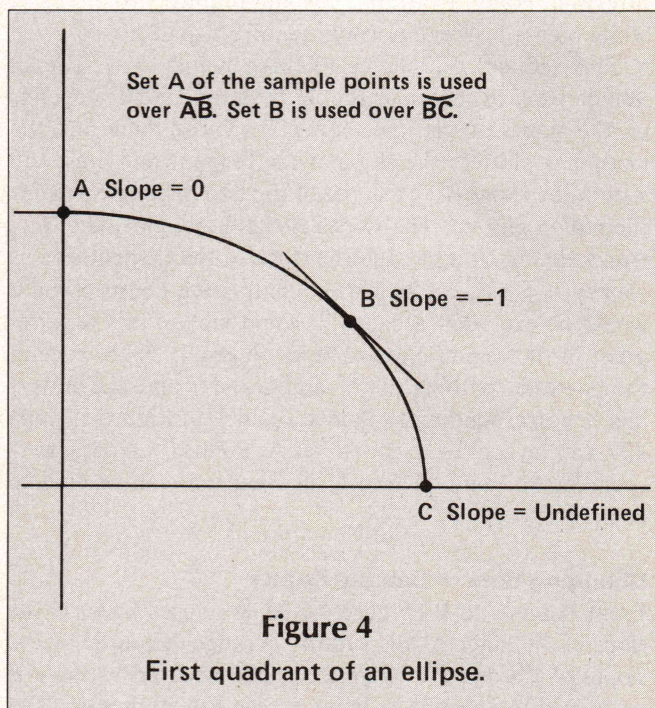
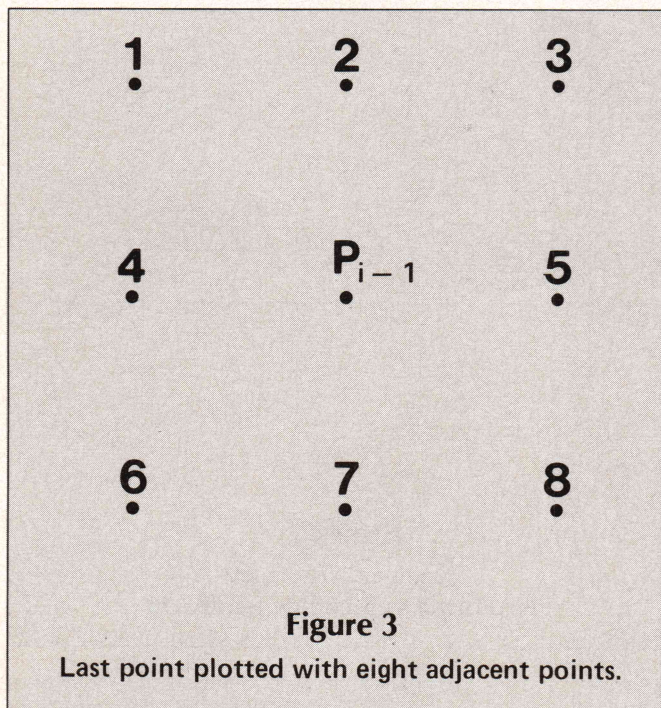
This algorithm takes advantage of the symmetry of an ellipse. The ellipse is symmetric about the x and y axes, and about its center. Therefore, an ellipse need only be mapped in the first quadrant. The other three quadrants

are easily plotted.

Clipping is done by the routine in Listing Two (page 48). The algorithm assumes medium resolution. DDJ

(Listings begin on page 45)

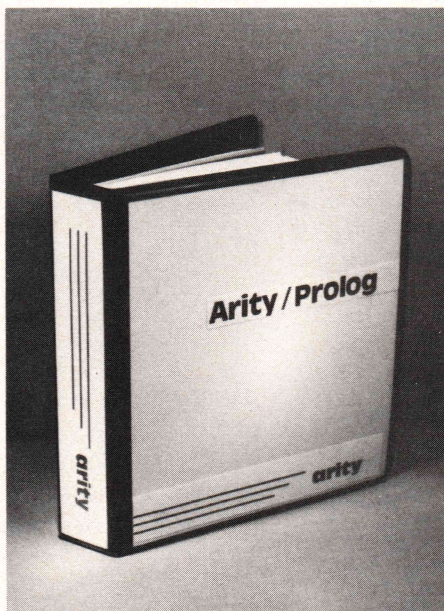
Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service No. 193.



More Power Than You Thought Possible

Arity offers the first serious implementation of Prolog for IBM personal computers. Arity/Prolog is a powerful, highly optimized, and extended version of the logic programming language Prolog. Imagine building software applications with a language that solves problems through deduction and logical inference. The task of creating complex programs is much faster and easier, resulting in lower development costs. Arity/Prolog is now in use in a wide range of applications in industry, business, research, and education. The solution—the *Arity/Prolog Interpreter*:

- Source level debugger
- Virtual databases, each with a workspace of 16 megabytes
- Floating-point arithmetic
- String support for efficient text handling



- Interface to assembly language and 'C'
- Text screen manipulation
- Integrated programming shell to MSDOS
- Comprehensive set of evaluable predicates
- Definite clause grammar support

Arity/Prolog Interpreter \$495.00

Arity also offers the *Arity/Prolog Compiler and Interpreter*, a sophisticated development environment for building AI applications. Essential for producing fast, serious production code.

Arity/Prolog Compiler and Interpreter \$1950.00

The *Arity/Prolog Demo Disk* is available for \$19.95. ■ Arity/Prolog products run on the IBM PC, XT, AT, and all IBM compatibles. ■ To order, call (617) 371-2422 or use the order form below.

arity corporation 358 Baker Avenue, Concord, MA 01742

Name _____

Organization _____

Address _____

- ☐ Enclosed is a check or money order to Arity Corporation

[illegible]

Valid ____/____/____ to ____/____/____ signature _____

Quantity	Product	Unit Price	Total Price
	Arity/Prolog Compiler & Interpreter	\$1950.00	
	Arity/Prolog Interpreter	\$ 495.00	
	Arity/Prolog Demo Disk	\$ 19.95	
Subtotal			
MA residents add 5% sales tax			
Total Amount			

- ☐ Please send me more information about Arity and Arity/Prolog

arity 358 Baker Avenue, Concord, MA 01742

M-AD-01

The well known equation of an ellipse is:

$$\alpha Y^2 + \beta X^2 - \alpha\beta = 0. \quad \text{Eq. 1.}$$

Therefore define an error term $D(P_i)$ such that at any sample point P_i along the curve:

$$D(P_i) = \alpha Y^2 + \beta X^2 - \alpha\beta. \quad \text{Eq. 2}$$

Define two points S_i and T_i relative to P_{i-1} such that P_i will be selected from one of these two points (see Figure 5).

Define a decision variable d_i such that:

$$d_i = D(S_i) + D(T_i). \quad \text{Eq. 3}$$

Notice that if the curve passes midway between S_i and T_i , $d_i = 0$. Further notice that:

- 1) if the curve passes above the midpoint, $d_i < 0$, and
- 2) if the curve passes below the midpoint, $d_i > 0$.

Therefore:

- 1) if $d_i < 0$, we choose S_i , and
- 2) if $d_i \geq 0$, we choose T_i .

The quantity to add to d_i , depending on whether S_i or T_i is chosen, must be determined. The value of d_i for S_i and T_i must first be determined:

$$\begin{aligned} d_i &= D(S_i) + D(T_i) \\ d_i &= \alpha Y_{i-1}^2 + \beta(X_{i-1} + 1)^2 - \alpha\beta + \alpha(Y_{i-1} - 1)^2 \\ &\quad + \beta(X_{i-1} + 1)^2 - \alpha\beta. \\ d_i &= \alpha Y_{i-1}^2 + \alpha(Y_{i-1} - 1)^2 + 2\beta(X_{i-1} + 1)^2 \\ &\quad - 2\alpha\beta. \end{aligned} \quad \text{Eq. 4}$$

Determine d_{i+1} if S_i is chosen:

$$\begin{aligned} d_{i+1} &= \alpha Y_{i-1}^2 + \beta(X_{i-1} + 2)^2 - \alpha\beta + \alpha(Y_{i-1} - 1)^2 \\ &\quad + \beta(X_{i-1} + 2)^2 - \alpha\beta. \\ d_{i+1} &= \alpha Y_{i-1}^2 + \alpha(Y_{i-1} - 1)^2 + 2\beta(X_{i-1} + 2)^2 \\ &\quad - 2\alpha\beta. \end{aligned} \quad \text{Eq. 5}$$

Define Δ_s , the difference between d_{i+1} and d_i if S_i is chosen:

$$\begin{aligned} \Delta_s &= d_{i+1} - d_i \\ \Delta_s &= 2\beta(X_{i-1} + 2)^2 - 2\beta(X_{i-1} + 1)^2. \\ \Delta_s &= 2\beta(2X_{i-1} + 3). \\ \Delta_s &= 4\beta X_{i-1} + 6\beta. \end{aligned} \quad \text{Eq. 6}$$

Now calculate the quantity to be added to d_i if T_i is chosen:

$$d_{i+1} = \alpha(Y_{i-1} - 1)^2 + \beta(X_{i-1} + 2)^2 - \alpha\beta$$

$$\begin{aligned} d_{i+1} &= \alpha(Y_{i-1} - 1)^2 + \beta(X_{i-1} + 2)^2 - \alpha\beta \\ &\quad + \alpha(Y_{i-1} - 2)^2 + \beta(X_{i-1} + 2)^2 - \alpha\beta. \end{aligned}$$

$$\begin{aligned} d_{i+1} &= \alpha(Y_{i-1} - 1)^2 + \alpha(Y_{i-1} - 2)^2 \\ &\quad + 2\beta(X_{i-1} + 2)^2 - 2\alpha\beta. \end{aligned} \quad \text{Eq. 7}$$

Define Δ_T , the difference between d_{i+1} and d_i when T_i is chosen:

$$\begin{aligned} \Delta_T &= d_{i+1} - d_i \\ \Delta_T &= 2\beta(X_{i-1} + 2)^2 - 2\beta(X_{i-1} + 1)^2 \\ &\quad + \alpha(Y_{i-1} - 2)^2 - \alpha Y_{i-1}^2. \\ \Delta_T &= 4\beta X_{i-1} + 6\beta - 4\alpha Y_{i-1} + 4\alpha. \end{aligned} \quad \text{Eq. 8}$$

Now the value of d_i for the initial point, which is located on the Y-axis, must be determined. R_m is half the length of the major axis. If the Y-axis is the major axis, the initial point is the point $P(0, R_m)$. Therefore:

$$d_i = 2\alpha R_m^2 - 2\alpha R_m + \alpha + 2\beta - 2\alpha\beta. \quad \text{Eq. 9}$$

If the X-axis is the major axis, the initial point is the point $P(0, (\beta/\alpha) \times R_m)$. Therefore:

$$d_i = (2\beta^2/\alpha) \times R_m^2 - 2\beta R_m + \alpha + 2\beta - 2\alpha\beta. \quad \text{Eq. 10}$$

Consider now the slope of a line tangent to the ellipse. Beginning with the original point, it can be guaranteed that the slope of the tangent line will always decrease while the ellipse is being mapped in the first quadrant. Therefore, once the slope of the tangent line reaches -1 , it will never be greater than -1 . Also, it may be seen that from the point where the slope of the tangent line reaches -1 throughout the rest of the first quadrant, for any point P_{i-1} mapped, T_i will be closer than S_i . At some point, the ellipse will diverge enough from the tangent line of slope -1 for a point U_i (see Figure 6) to be closer to the ellipse than T_i . It will be sufficient to check the new set of points (T_i and U_i) from the point where the tangent line's slope becomes -1 . d_i must be recalculated, as must Δ_T and Δ_U . Calculate d_i :

$$\begin{aligned} d_i &= D(T_i) + D(U_i) \\ d_i &= \alpha(Y_{i-1} - 1)^2 + \beta(X_{i-1} + 1)^2 - \alpha\beta \\ &\quad + \alpha(Y_{i-1} - 1)^2 + \beta X_{i-1}^2 - \alpha\beta. \\ d_i &= 2\alpha(Y_{i-1} - 1)^2 + \beta X_{i-1}^2 \\ &\quad + \beta(X_{i-1} + 1)^2 - 2\alpha\beta. \end{aligned} \quad \text{Eq. 11}$$

Determine d_{i+1} if T_i is chosen:

$$\begin{aligned} d_{i+1} &= 2\alpha(Y_{i+1} - 2)^2 + \beta(X_{i-1} + 1)^2 \\ &\quad + \beta(X_{i-1} + 2)^2 - 2\alpha\beta. \end{aligned} \quad \text{Eq. 12}$$

Define Δ_T as the difference between d_i and d_{i+1} if T_i is chosen:

Table
Derivation of the algorithm to map an ellipse.

$$\Delta_T = d_{i+1} - d_i.$$

$$= 2\alpha(-2Y_{i-1} + 3) + \beta(4X_{i-1} + 4).$$

Eq. 13

Determine d_{i+1} if U_i is chosen:

$$d_{i+1} = 2\alpha(Y_{i-1} - 2)^2 + \beta X_{i-1}^2$$

$$+ \beta(X_{i-1} + 1)^2 - 2\alpha\beta.$$

Eq. 14

Define Δ_U as the difference between d_{i+1} and d_i if U_i is chosen:

$$\Delta_U = d_{i+1} - d_i.$$

$$= 2\alpha(-2Y_{i-1} + 3).$$

Eq. 15

The point where the slope becomes less than -1 must be determined. Finding the first derivative of the ellipse produces the equation:

$$\alpha Y Y' = -\beta X.$$

Eq. 16

When the slope is required to be less than -1 , the relation

$$\alpha Y < \beta X$$

Eq. 17

determines the point at which this occurs. This concludes the derivation of the algorithm for mapping an ellipse.

Decision Variables (Text begins on page 40)

Listing One

```

/*  compiled 11/28/84 under Lattice C Compiler ver. 2.13
*
*  ellipse.c
*    by Tom Hogan
*  Version 2.0
*  Copyright 1984
*  by C Source Inc.
*  All rights reserved.
*  Permission is granted
*  for unlimited
*  personal, non-
*  commercial use only.
*
*  For answers to questions,
*  please write to:
*
*  Tom Hogan
*  C Source Inc.
*  12801 Frost Road
*  Kansas City, MO 64138
*
*  If you would rather call,
*  our number is (816) 353-8808.
*
*  Please call between 9 AM
*  and 5 PM CST, Monday thru
*  Friday. Calls can only
*  be returned on a collect basis.
*/

```

```

#define ERROR 0
#define SUCCESS 1

```

```

#define MAX_COL 319
#define MAX_ROW 199
#define MIN_COL 0
#define MIN_ROW 0

```

```

/* a macro used in the error-checking code */

```

```

#define inrange(a,x,b) ((a) <= (x) && (x) <= (b))

```

```

/* the bounds of the following array can be set for different
* graphics modes; this array is used in the assembly algorithm
* Listing 2 to write individual pixels.
*/

```

(Continued on next page)

Decision Variables (Listing Continued, text begins on page 40)

Listing One

```

int CRT_BNDS[] = { MAX_COL, MAX_ROW, MIN_COL, MIN_ROW };

    /* r_sub_m is half the length of the major axis */

ellipse(x, y, r_sub_m, color, aspect)
    int x, y, r_sub_m, color;
    double aspect;
{
    int two_x, two_y, or_color;
    int col, row, rel_x, rel_y; /* rel_x and rel_y are relative */
    double square_aspect;      /* to the center */
    long temp, beta, alpha, two_alpha, four_alpha, two_beta, four_beta, d;

        /* d is the decision variable */

        /* error checking follows */

    if ((aspect < 0.004) || !inrange(0, color, 15) || !inrange(1, r_sub_m,
        154)) {
        printf("ELLIPSE(%d, %d, %d, %d, %f) - BAD ARG\n", x, y, r_sub_m,
            color, aspect);
        return ERROR; }
    square_aspect = aspect * aspect; /* initialize the beginning row */
    if (aspect < 1.0) { /* and set the values of */
        alpha = r_sub_m * r_sub_m; /* constants */
        beta = alpha * square_aspect;
        row = y + r_sub_m * aspect; }
    else {
        beta = r_sub_m * r_sub_m;
        alpha = beta / square_aspect;
        row = y + r_sub_m; }
    if (alpha == 0L) alpha = 1L; /* compensates for very small ellipses */
    if (beta == 0L) beta = 1L;
    or_color = 0xc00 | color; /* sets the high byte for a screen interrupt */
    col = x; /* initializes the beginning column */
    two_x = x << 1;
    two_y = y << 1;
    rel_y = row - y;
    two_alpha = alpha << 1;
    four_alpha = alpha << 2;
    four_beta = beta << 2;
    two_beta = beta << 1;

        /* initialize the decision variable -- Eq. 9 or 10 */

    d = two_alpha * ((rel_y - 1) * rel_y) + alpha + two_beta * (1 - alpha);

        /* when the slope <= -1, choose the new set of points */
        /* -- Eq. 17 */

    while (alpha * (rel_y = row - y) > beta * (rel_x = col - x)) {
        write_pix(col, row, two_x - col, two_y - row, or_color);
        if (d >= 0) {
            d += four_alpha * (1 - rel_y); /* Eq. 6; also first */
            row--; /* half of Eq. 8; */
            d += two_beta * (3 + (rel_x << 1)); /* second half of Eq. 8 */
            col++; }

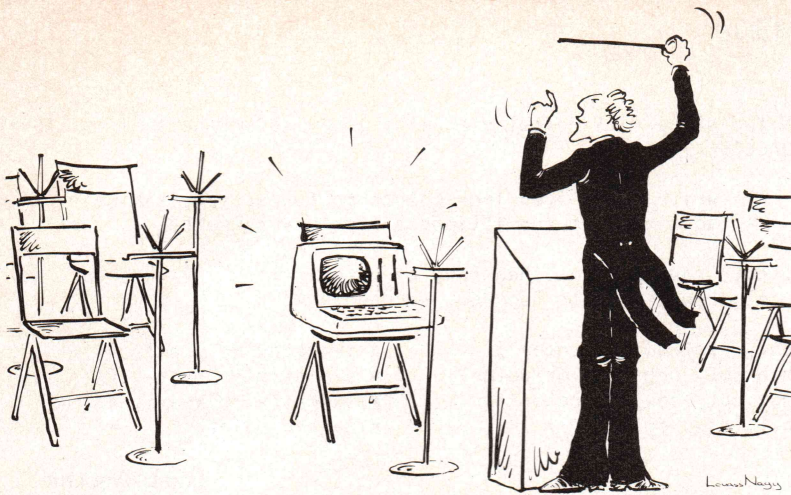
        /* initialize the decision variable for the rest of the ellipse */
        /* -- Eq. 11 */

    d = two_beta * (rel_x * (rel_x + 1) + two_alpha * (rel_y * (rel_y - 2)
        + 1) + (1 - two_alpha) * beta;

    while ((rel_y = row - y) + 1) {
        write_pix(col, row, two_x - col, two_y - row, or_color);
        if (d <= 0) { /* col - x == rel_x */
            d += four_beta * (1 + col - x); /* Eq. 15; also first */
            col++; /* half of Eq. 13; */
            row--;

```

(Continued on page 48)



Would you hire an entire band when all you need is one instrument? Of course not.

So why use a whole orchestra of computers when all you need is one to develop software for virtually any type of micro-processor?

The secret? Avocet's family of cross-assemblers. With Avocet cross-assemblers you can develop software for practically every kind of processor — *without having to switch to another development system along the way!*

Cross-Assemblers to Beat the Band!

Development Tools That Work

Avocet cross-assemblers are fast, reliable and user-proven in over 4 years of actual use. Ask NASA, IBM, Xerox or the hundreds of other organizations that use them. Every time you see a new micro-processor-based product, there's a good chance it was developed with Avocet cross-assemblers.

Avocet cross-assemblers are easy to use. They run on almost any personal computer and process assembly language for the most popular microprocessor families.

Your Computer Can Be A Complete Development System

Avocet has the tools you need to enter and assemble your software and finally cast it in EPROM:

VEDIT Text Editor makes source code entry a snap. Full-screen editing plus a TECO-like command mode for advanced tasks. Easy installation - INSTALL program supports over 40 terminals and personal computers. Customizable keyboard layout. CP/M-80, CP/M-86, MSDOS, PC DOS \$150

EPROM Programmers let you program, verify, compare, read, display EPROMS but cost less because they communicate through your personal computer or terminal. No personality modules! On-board intelligence provides menu-based setup for 34 different EPROMS, EEPROMS and MPUs (40-pin devices require socket adaptors). Self-contained unit with internal power supply, RS-232 interface, Textool ZIF socket. Driver software (sold separately) gives you access to all programmer features through your computer, lets you download cross-assembler output files, copy EPROM to disk.

Model 7228 Advanced Programmer — Supports all PROM types listed. Superfast "adaptive" programming algorithm programs 2764 in 1.1 minutes.

Model 7128 Standard Programmer — Lower-cost version of 7228. Supports all PROM types except "A" versions of 2764 and 27128. Standard programming algorithm programs 2764 in 6.8 minutes.

Avocet Cross-assembler	Target Microprocessor	CP/M-80	CP/M-86 IBM PC, MSDOS**
XASM04 <i>NEW</i>	6804	\$ 250.00	\$ 250.00
XASM05	6805	200.00	250.00
XASM09	6809	200.00	250.00
XASM18	1802/1805	200.00	250.00
XASM48	8048/8041	200.00	250.00
XASM51	8051	200.00	250.00
XASM65	6502/65C02	200.00	250.00
XASM68	6800/01, 6301	200.00	250.00
XASM75	NEC 7500	500.00	500.00
XASM85	8085	250.00	250.00
XASM400	COP400	300.00	300.00
XASMF8	F8/3870	300.00	300.00
XASMZ8	Z8	200.00	250.00
XASMZ80	Z80	250.00	250.00
XMAC682 <i>NEW</i>	68200	595.00	595.00
XMAC68K <i>NEW</i>	68000/68010	595.00	595.00

Model 7956 and 7956-SA Gang Programmers — Similar features to 7228, but program as many as 8 EPROMS at once. 7956-SA stand-alone version copies from a master EPROM. 7956 lab version has all features of stand-alone plus RS-232 interface.

EPROM: 2758, 2716, 2732, 2732A, 2764, 2764A, 27128, 27128A, 27256, 2508, 2516, 2532, 2564, 68764, 68766, 5133, 5143. **CMOS:** 27C16, 27C32, 27C64, MC6716. **EEPROM:** 5213, X2816A, 48016, I2816A, 5213H. **MPU (w/adaptor):** 8748, 8748H, 8749, 8749H, 8741, 8742, 8751, 8755.

7228	Advanced Programmer	\$ 549
7128	Standard Programmer	429
7956	Laboratory Gang Programmer	1099
7956-SA	Stand-Alone Gang Programmer	879
GDX	Driver Software	95
481	8748 Family Socket Adaptor	98
511	8751 Socket Adaptor	174
755	8755 Socket Adaptor	135
CABLE	RS-232 Cable (specify gender)	30

HEXTRAN Universal HEX File Converter — Convert assembler output to other formats for downloading to development systems and target boards. Also useful for examining object file, changing load addresses, extracting parts of files. Converts to and from Intel, Motorola, MOS, RCA, Fairchild, Tektronix, TI, Binary and HEX/ASCII Dump formats. For CP/M, CP/M-86, MSDOS, PC DOS \$250

Ask about UNIX.

68000 CROSS-ASSEMBLER — With exhaustive field testing completed, our 68000 assembler is available for immediate shipment. XMAC68K supports Motorola standard assembly language for the 68000 and 68010. Macros, cross-reference, structured assembly statements, instruction optimization and more. Linker and librarian included. Comprehensive, well-written manual.

To find out more, call us toll-free.

1-800-448-8500

(in the U.S. Except Alaska and Hawaii)

VISA and Mastercard accepted. All popular disc formats now available — please specify. Prices do not include shipping and handling — call for exact quotes. OEM INQUIRIES INVITED.

*Trademark of Digital Research **Trademark of Microsoft



Sales and Development: 585-DDJ
10 Summer Street
P.O. Box 490, Dept.
Rockport, Maine 04856
(207) 236-9055 Telex: 467210 AVOCET CI

Corporate Offices:
804 South State Street
Dover, Delaware 19901

REMOVE



from your **C** programs
with

PC-LINT

PC-LINT analyses your C programs (one or many modules) and uncovers glitches, bugs, quirks and inconsistencies. It will catch subtle errors before they catch you.

PC-LINT resembles the Lint that runs on the UNIX O.S. but with more features and greater sensitivity to the problems of the 8086 environment.

- Full K&R C
- Supports multiple modules—finds inconsistencies between declarations and use of functions and data across a set of modules comprising a program.
- Compares function arguments with the associated parameters and complains if there is a mismatch or too many or too few arguments.
- All warning and information messages may be turned on and off globally or locally (via command line and comments) so that messages can be tailored to your programming style.
- All command line information can be furnished indirectly via file(s) to automate testing.
- Use it to check existing programs, programs about to be exported or imported, as a preliminary to compilation, or **prior** to scaling up to a larger memory model.
- All one pass with an integrated pre-processor so it's very fast.
- Has numerous flags to support a wide variety of C's, memory models, and programming styles.
- **Price: \$98.00 MC, VISA**
(includes shipping and handling) PA residents add 6% sales tax. Outside USA add \$10.00.
- Runs on the IBM PC (or XT, AT or compatible) under DOS 2.0 and up, with a minimum of 128KB of memory. It will use all the memory available.

GIMPEL SOFTWARE

3207 Hogarth Lane • Collegeville, PA 19426
(215) 584-4261

*Trademarks: IBM (IBM Corp.), PC-LINT (Gimpel Software), UNIX (AT&T)

Decision Variables (Listing Continued, text begins on page 40)

Listing One

```
d += two_alpha * (3 - (rel_y << 1)); } /* second half of Eq. 13 */
return SUCCESS; }

/* write_pix writes four points to the screen, taking *
 * advantage of the ellipse's four-way symmetry */

write_pix(col, row, neg_col, neg_row, or_color)
int col, row, neg_col, neg_row, or_color;
{
    _pixasm(col, neg_row, or_color); /* symmetry -- y axis */
    _pixasm(neg_col, neg_row, or_color); /* symmetry -- center */
    _pixasm(neg_col, row, or_color); /* symmetry -- x axis */
    _pixasm(col, row, or_color); /* mapped point */
}
```

End Listing One

Listing Two

8086 Assembly function that clips against the screen bounds

```

;          _pixasm(x, y, type_color)
;          int x, y, type_color
;          type - get or read pix in hi byte
;          color - if set pix, in lo byte
;
DGROUP GROUP DATA
DATA SEGMENT WORD PUBLIC 'DATA'
ASSUME DS:DGROUP

EXTRN CRT_BNDS:WORD

PGROUP GROUP PROG
PROG SEGMENT BYTE PUBLIC 'PROG'
ASSUME CS:PGROUP

PUBLIC _pixasm

_pixasm proc near
    push bp ; set return address
    mov bp, sp

    mov cx, [bp+4] ; set x (column)
    cmp cx, CRT_BNDS ; check for max x value
    jg BYE
    cmp cx, CRT_BNDS+4 ; check for min x value
    jl BYE
    mov dx, [bp+6] ; set y (row)
    cmp dx, CRT_BNDS+2 ; check for max y value
    jg BYE
    cmp dx, CRT_BNDS+6 ; check for min y value
    jl BYE
    mov ax, [bp+8] ; set color (al) / fcn number (ah)
    int 10h ; do video interrupt
    sub ah, ah
    BYE: pop bp ; reset frame pointer
    ret ; bye!

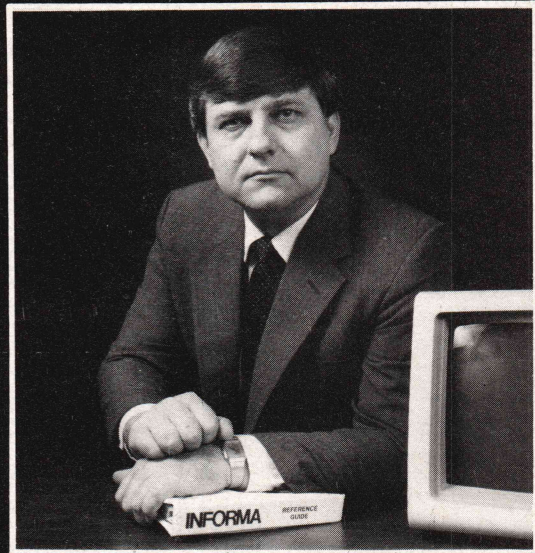
_pixasm endp
PROG ends
DATA ends
end
```

End Listings

"It's Time Someone Got Serious About The CRITICAL Issues Concerning LAN Software"

"The Industry has a responsibility to the buying public to stop using MIRRORS to promote VAPORWARE claims about multi-user software capabilities. Many complex and potentially DANGEROUS situations exist for any software that runs in a multi-user / LAN environment. Partial solutions may be acceptable when using word processing or spreadsheet applications, however Data Base Management Systems demand complete solutions. Unlimited Processing Incorporated has devoted millions of dollars and three and one half years developing the expertise necessary to present a true interactive multi-user DBMS and APPLICATION DEVELOPMENT ENVIRONMENT.

George Treiber
President, UP, Inc.



To evaluate "multi-user claims," you must ask the following questions:

- **IS IT REALLY MULTI-USER, NOT MERELY RESTRICTED SINGLE-USER?**

INFORMA Is!

Can an unlimited number of simultaneous users:

- Access the same file?
- Access the same file in different sort orders?
- Access the same record?
- Update the same record?
- Enter new records into the same file?

Does the system prevent one user from changing the appearance of the database while another user is reviewing that data?

- **DOES IT REALLY MANAGE YOUR DATABASE, NOT JUST STORE DATA?**

INFORMA Does!

With complete transparency to users does it:

- Allow data files to reside on different disk drives?
- Keep track of the existence and location of data files for all applications?
- Eliminate data compression and other time consuming operations?
- Permit data file movement for disk balance?

- **IS DATA INTEGRITY GUARANTEED, INSTEAD OF ACCEPTING DATA LOSS AS AN INEVITABLE RISK?**

INFORMA Guarantees!

Can you be assured one update will not overwrite another when two or more users simultaneously:

- Change the same record?
- Change adjacent records?

- **IS COMPLETE SECURITY EMBEDDED THROUGHOUT THE SYSTEM, NOT MERELY AN AFTERTHOUGHT?**

It Is With INFORMA

- Are multiple levels of user security offered?
- Can each activity be secured by specific user?
- Can applications be secured by specific user?
- Can audit trails be created by user and application?
- Can the functions find, enter, update, delete, and print be secured by user and application?

- **IS IT FAST ENOUGH TO BE EFFECTIVE, NOT TOO SLOW TO BE USEFUL?**

INFORMA Has Unequaled Speed!

Retrieve any record in less than 1 sec?

Does the access time remain relatively constant:

- As the number of records increases?
- As the number of users increases?

INFORMA offers the only serious solution to Interactive, Real-Time, DataSharing.



**UNLIMITED PROCESSING
INCORPORATED**

8382 Baymeadows Road, Suite 8
Jacksonville, Florida 32216
(904) 731-8330 and (800) 874-8555
Telex 350754 (800) 874-4185

Introductory Offer

Single-user

\$199

regularly \$795

LAN/Multi-user

\$599

regularly \$1495

Offer Expires April 10, 1985

Circle no. 87 on reader service card.

Solid Shape Drawing on the Commodore 64

by Richard Rylander

Computer graphics is one of the fastest growing areas of computer software and hardware development. The old saying that "one picture is worth a thousand words" couldn't be truer when you consider the effectiveness of a pie chart or bar graph versus columns of numbers. Even the simplest of home computers is capable of at least this basic form of computer graphics. But the term *computer graphics* usually brings to mind more exciting images than just mundane business applications.

Special visual effects for motion pictures and some television advertisements are now being done with computers to create scenes of "enhanced reality." We see spacecraft making impossible maneuvers or shimmering corporate logos "flying

easy-to-use graphics capabilities to the masses. The "masses," however, may still find the Macintosh pricey, especially if they just want to play around a bit with MacPaint. Low-cost computers, such as the Commodore 64, while lacking the slick graphic interaction of the Mac, are selling for such ridiculously low prices (\$150 at the time of writing this article) that they are hard to pass up. They have the potential for some fairly sophisticated graphics, but so far most of the commercially available software has done little more than add the usual "point plotting" and "line drawing" commands.

This article describes a software package that will allow you to create your own computer-generated scenes, such as "Coffee and Donuts" in Figure 1 (page 55) and "Goblets"

Pumping pixels requires tight, efficient code. Here are halftone shading, backlighting and other visual effects in a tight little package.

by" in ways that would be difficult to simulate with conventional photographic techniques. The enormous amount of data and processing necessary to produce such images has made the use of "super computers" essential. Even with the largest computers working full time on the task, only a few feet of film are produced per day. For home computer owners who want to create their own computer-generated masterpiece, however, the task is not hopeless.

The Apple Macintosh represents a major step in bringing powerful,

in Figure 2 (page 55). You can construct images from combinations of elementary shapes (spheres, cylinders, and toroids in various orientations) or by the "polygon mesh" technique typically used to render more complex objects. You can produce drawings with realistic shading effects and in a variety of styles to make some surprisingly detailed pictures with a minimum of effort. The package includes demonstration programs to illustrate how you use each graphic function and style option.

In keeping with the "running light without overbyte" theme of *DDJ*, the entire graphics package fits in 3K of RAM. The program sits in an area of

Richard Rylander, 179 N. McKnight Rd. Apt. 203, St. Paul, MN 55119.

RAM that is inaccessible to BASIC (a 4K block following the BASIC ROM) so that you don't lose any useful program space. Commodore supplies a "DOS Wedge" program that occupies the last 1K of this block, and one of my objectives in the design of the graphics package was to maintain compatibility with this useful utility.

The compactness of the code required me to leave out some "bells and whistles," but I've made no sacrifices to execution speed or user-interface ease. The main omission is error trapping—the software performs no checking to ensure that you use only "legal" point coordinates and so on. This may make program development a little more difficult, but once you have written and debugged a program properly, error checking becomes extraneous and only slows program execution.

We must address several general tasks in developing a shape-drawing program. The first is, of course, how to calculate the apparent brightness for all points on visible surfaces of objects. Next, how do we display these different brightness levels on a high-resolution display that is basically on or off? Finally, a "general" task, but detailed in nature, is creating the specific software tools that will let us do the required calculations and bit-map manipulations.

The program itself is written in machine language, because even with our simplifying restrictions, the plotting of each pixel still involves considerable computation. The program uses integer arithmetic throughout, and we utilize all the symmetry available to eliminate redundant calculations. The main program actually consists of five separate subprograms to break the process into manageable pieces and to provide a library of separate utilities that you may find useful in other programs. Before getting into the problem of shade calculation, I'll present the first subprogram, which is a collection of integer arithmetic utilities that all the later subprograms need.

Integer Arithmetic Utilities

Listing One (page 61) provides the source code for a set of four integer

arithmetic routines. Because we will draw our shaded shapes on a high-resolution display of 320 x 200 points, we need only single-precision arguments for most functions and double-precision results for some intermediate values.

The first two routines, multiplying two single-precision numbers to yield a double-precision product and dividing a double-precision dividend by a double-precision divisor to yield a double-precision quotient, are fairly standard routines that should need no description other than the comments

in the code. The multiplication routine also contains a special case to treat a single-precision number as a signed integer, which is then squared.

The integer-square-root routine deserves some special comment. Have you ever tried the BASIC square-root function? If you have, you know it takes about 52 milliseconds. At this rate, plotting the 64,000 pixels of a 320 x 200 screen would take about 3,328 seconds, or nearly an hour, if we need a square root per point! Of course, we don't need a square root for each point, but it is an

Advanced Screen Management made easy

Now a professional software tool from
Creative Solutions.

WINDOWS FOR C™

More than a window display system,
WINDOWS FOR C is a video tool kit for all
screen management tasks.

- Pop-up menus and help files
- Auto memory management
- Keyboard interpreter
- Word wrap
- Auto scroll
- Highlighting
- Color control
- Overlay and restore
- Plus a library of over 50
building block subroutines

**Designed for enhanced portability.
Easy to learn, easy to use.**

Once you've tried WINDOWS FOR C,
you'll wonder how you ever managed without it.

Full support for IBM PC/XT/AT and compatibles, plus interfaces for non-IBM computers:
Lattice C, CI-C86, Mark Wm. C, Aztec C, Microsoft C, DeSmet C (PC/MSDOS),

NEW Ver. 3.1

Enhanced portability.
Topview compatible.

WINDOWS FOR C \$195
(specify compiler & version)

Demo disk and manual \$ 30
(applies toward purchase)

**Full source available.
No royalties.**



Creative Solutions

21 Elm Ave., Box T4,
Richford, VT 05476

802-848-7738

Master Card & Visa Accepted
Shipping \$2.50
VT residents add 4% tax.

Circle no. 27 on reader service card.

essential part of many shade calculations. Obviously, anything we can do to speed up this particular function will have a great effect on the overall program speed.

The BASIC square-root routine is slow because BASIC is written to save space, not time. Virtually all small BASICs find a square root by first taking the log of the argument, dividing that result by 2, and then exponentiating. Because LOG and EXP functions are already available, why use any more space than necessary to derive SQR?

One way to speed up this function is to use Newton's method, an iterative procedure that can give full floating-point precision in less time. Because we are interested only in integer results, we can do much better yet, though.

The method we actually use is essentially as fast as doing a single-divide operation. We construct the root, bit by bit, in a manner similar to the way we would take a decimal square root by hand using pencil and paper (does anyone remember how to do that in this age of electronic calculators?). The process is even easier for a binary root because the guesses are, of course, only 1 or 0. The procedure is almost identical to that for division: we guess a partial root bit (as we would guess the next quotient bit) and, in effect, compare the square of the partial quotient to the argument to see if we keep the 1 guessed or change it to a 0 and restore the argument. For more details about this procedure, I recommend section 17.3, "Binary Square Roots—Restoring Method," of the book *The Logic of Computer Arithmetic* by Ivan Flores (Englewood Cliffs, NJ: Prentice-Hall).

The last integer arithmetic routine is a fast procedure for generating pseudo-random bytes. An old and popular method for generating pseudo-random numbers is the so called congruential method, in which you derive each successive random number from the previous random number by multiplying by a suitably chosen constant and taking the product modulo P, where P is a large prime (see, for example, "A Better Random Number Generator," by H. Cem

Kaner and John R. Vokey in the June 1984 issue of *MICRO*). Again, though, because we are interested only in generating random bytes, we can use a much faster routine.

The method we use is to store two previous random bytes, exclusive OR them bit by bit, and rotate the result to the right by one bit to generate a new random byte. Because it needs no multiply or divide operations, provides a fairly uniform distribution of random bytes, and has a long period, this process is fast. Note that even though we are only generating random bytes (0-255), the pattern or sequence of successive bytes doesn't repeat until after 35,805 calls to the random-number-generator routine (for the initial parameters used in the program). The long period is necessary to ensure that we won't produce any unwanted secondary patterns in what should be "randomly" shaded images—the human eye is adept at picking out such correlations.

Now that some low-level number crunching is out of the way, we can concentrate on the graphics aspects of the main program.

Graphics Utilities

Listing Two (page 62) provides the source code for the elementary graphics functions of displaying and clearing the bit-map, filling the color map (determining dot/background colors), and plotting (and unplotting) individual points both directly and with the plot or unplot decision weighted by a shade value and a particular shade style function. The PLOT (and UNPLOT), CLEAR, COLOR, GRFON (display graphics screen), and GROFF (return to text screen) routines are specific to the Commodore 64. To adapt this program for other 6502-based computers, you need to rewrite these routines, but the rest of the graphics package (save the final user interface to BASIC) is machine independent.

The SHADE routine is simple and straightforward. We set up an 8 x 8 threshold matrix as a linear array of values 0 to 63. The six bits we need to address an element of this threshold array are determined by masking off the lower three bits of the absolute X

and Y screen coordinates (we take X modulo 8 and Y modulo 8) with the Y bits shifted to become the upper three bits. This effectively repeats the threshold pattern over the entire bit-map area. For each point within an object to be drawn, we calculate a shade value normalized to the range 0-63 and then compare it to the threshold value at that screen point to decide whether to plot or unplot.

If we have, for example, a large area where the shade value is constant at 31 (a 50 percent gray), then within each 8 x 8-pixel character cell block, half of the threshold values will be greater than or equal to the shade value, turning on half the pixels. The order in which the values 0 to 63 are arranged in the threshold table determines how these on pixels are distributed. A common pattern for these so called ordered-dither matrices is a recursive arrangement such as that shown in Table 1 (page 53). If that matrix is subdivided into quarters, sixteenths, and so on, each submatrix has the same general pattern, with offsets between submatrices.

The matrix in Table 1 (classic Bayer ordered dither) keeps the on and off pixels spread as far apart as possible for any shade. This keeps the spatial frequencies in the shade "texture" as high as possible for a particular shade. A disadvantage, though, is that texture changes then accompany shade changes, making shade quantization more apparent. Another disadvantage is that many color monitors have a hard time coping with the very high frequency on-off-on sequence of pixels, distorting shade values when isolated pixels get lost.

As an aside, a convenient algorithm to generate the ordered-dither matrix in Table 1 (or such a matrix of any size) is found in Figure 7 (page 59).

The threshold matrix we actually use in the program is shown in Table 2 (page 53). Here we follow a recursive scheme as we start turning on pixels until eight nuclei have been established. As shades increase, new pixels are added around the edges of these nuclei, simulating the dot-growth behavior seen in normal printing halftones. Except in the ex-

treme highlight and shadow regions, the shade texture remains fairly constant. Also, the clustering of on or off pixels is much less demanding on the display bandwidth. But take your pick—try filling the threshold matrix with your own patterns.

The RSHADE routine shades by comparing the shade value to a pseudo-random byte shifted right twice to match the 0-63 range. This scheme also tends to average out the tone errors generated as each pixel is turned only on or off (though we want an intermediate shade of gray) by dithering the threshold value randomly at each pixel. Both shading schemes produce sharp edges because each pixel is plotted independently. Abrupt shade changes are then followed faithfully.

The SCALE routine provides an opportunity to mention a general rule you should follow when you try to write fast programs: If at all possible, avoid division; and if you must divide, try to make it by a power of 2 (so that you can use right shifts). The SCALE routine helps correct some of the geometric distortion that otherwise results from plotting objects on the Commodore 64's rectangular bit map. While the monitor display has an aspect ratio of approximately 4:3, the bit-map aspect ratio is 320:200 or 8:5. A simple way to keep sphere outlines circular (instead of the usual egglike appearance) is to work in pseudoscreen coordinates of 320×240 , giving the bitmap the same 4:3 aspect ratio as the screen display. The SCALE routine then converts 0-239 YPLOT values to a 0-199 range.

An obvious way to do this is to first multiply by 5 and then divide by 6. Or you might save a multiplication by first dividing a copy of YPLOT by 6 and subtracting that from the original YPLOT. A *much* faster way is to multiply the single-precision YPLOT by 213 and then take just the upper byte of the double-precision product (effectively dividing by 256) as the scaled YPLOT value. This gives us the proper range of absolute screen Y values and "rounds" the scaling as well.

This method of scaling means that you effectively replot every sixth Y line of pixels, but the routine's sim-

plicity more than makes up for plotting 20 percent more points. Scaling is left as an option because some printers (such as the Commodore 1525) have 1:1 dot densities. By not scaling, the screen display is distorted but a hard-copy printout will have the proper geometry. On the other hand, you can set up printers, such as the Epson RX-80, with the appropriate horizontal and vertical dot densities to produce a 4:3 aspect ratio screen dump. Scaling here corrects both the screen and the hard-copy output.

The routine PLTSHD is a higher level routine that simply checks a couple of flags to see what kind of shading we want and whether to scale or not. It then calls the appropriate shading routine. To plot a shade-weighted pixel from BASIC, POKE a shade value (0-63) into VALUE; POKE the absolute X and (optionally scaled) Y values into XPLOT, XPLOT+1, and YPLOT; set up the flags HTORRN (HALFTONE or RANDOM) and NOSCAL; then

SYS to PLTSHD. This seems like a lot of work to plot a single point, but we really won't be plotting shade-weighted points by hand—the later shape-drawing routines take care of this.

Line and Facet Drawing

Listing Three (page 64) completes the elementary graphics functions by adding line-drawing and shaded-facet-drawing routines. We draw lines using a modified form of Bresenham's algorithm, a DDA (Digital Differential Analyzer) technique that keeps the actual plotted points within one half-screen unit of the true line. Regardless of the order in which we specify the line's endpoints, the program sorts them so that lines are always drawn from left to right (the X position is incremented or unchanged at each step). We then need only determine which is greater, the change in X or the change in Y, and whether the Y coordinate difference is positive or negative. The program checks

0	32	8	40	2	34	10	42
48	16	56	24	50	18	58	26
12	44	4	36	14	46	6	38
60	28	52	20	62	30	54	22
3	35	11	43	1	33	9	41
51	19	59	27	49	17	57	25
15	47	7	39	13	45	5	37
63	31	55	23	61	29	53	21

Table 1
Recursive Arrangement of Ordered-Dither Matrix

0	8	53	61	2	10	55	63
16	24	37	45	18	26	39	47
49	57	4	12	51	59	6	14
33	41	20	28	35	43	22	30
3	11	54	62	1	9	52	60
19	27	38	46	17	25	36	44
50	58	7	15	48	56	5	13
34	42	23	31	32	40	21	29

Table 2
Threshold Matrix

the scale flag to see if the endpoints should be adjusted (in their Y coordinates) before plotting commences. This keeps a common coordinate system for drawing shaded shapes (spheres and so on) and lines in the same image. A flag MODE determines whether the line is drawn by setting or clearing pixels ("black" or "white" lines).

The program's last elementary function is to draw shaded triangular facets. A powerful and flexible technique for rendering objects is by means of a polygon mesh. Natural polyhedral objects (for example, cubes and pyramids) are obvious candidates for this method, but you can also approximate curved surfaces by a connected mesh of planar polygonal sections. As in piecewise-linear approximations of curves, the finer the segmentation, the better the approximation—although at the price of greater computational overhead. The coordinates of the polygon vertices then constitute a data base that you can manipulate easily for rotational transformations, perspective transformations, and so on. Triangular sections are the simplest to handle and the most general because you can break any higher-order polygon down into triangular sections.

As in the line-drawing routine, the program sorts the endpoints of the facets into a left-to-right order and checks the scale flag to maintain a consistent coordinate system. At each X position across the facet, a top and bottom Y coordinate pair is determined by a simple proportionality between the X difference and Y difference for the particular triangular sides. Although this involves both multiplication and division operations, it is not the innermost loop here (drawing the shaded line segment between Y pairs is). It also does not give more than one Y value per X position, as would a DDA method (for lines with slope greater than 1 in magnitude).

The maximum X difference is restricted to less than 256 (single precision), though the absolute X coordinates for the triangle vertices can be anywhere on the screen. This is not too severe a constraint because most polygon meshes are made up of small

sections, and, if a larger triangle is essential, you can break it into smaller triangles that meet this condition. The occasional inconvenience this might cause is worth the increase in speed and shortened program.

We leave the shade value used in drawing facets as a parameter that the calling program specifies. Although adding "surface normal" calculation and shade value computation based on some illumination model (and requiring the Z coordinates then to be specified for the vertices) is not difficult, the same shade value is used for all points of the facet, making shade determination by a BASIC program practical. Also, leaving shade calculations up to BASIC adds the flexibility that we can use any shading model. The curved surface-drawing routines I describe later do include shade determination, but here each point can potentially have a different shade value, putting it in the innermost loop. For this reason, the curved surfaces must have a fixed shading model.

An interesting use for the BASIC-specified shade values is drawing "white" facets by setting the shade to 64. This doesn't seem particularly useful, but when you draw "wire-frame" polyhedra, it makes a simple, hidden-line removal scheme possible. If the facets of an object are sorted so that they are drawn from those furthest to those nearest the observer, with "black" lines added around the edges of each facet, then foreground facets that partially obscure background facets erase the lines in their interior. We can set a flag EDGES so that, after the program has drawn a shaded facet, it can add lines automatically to outline and emphasize the edges. The MODE flag again determines how the lines are drawn (set or cleared).

Now that we have a toolbox of arithmetic and primitive-graphics routines, we can concentrate on the main subprogram for drawing shaded, curved surfaces rapidly.

Determining Shade Values for Curved Surfaces

Many recent advances in computer image synthesis have dealt specifical-

ly with how different surfaces reflect, refract, scatter, or absorb light. Increasingly complex models for the visual appearance of various surface textures have led to ever more realistic images, but increasing computational overhead accompanies these models. We'll employ the simplest shading scheme to start with and then suggest how you might modify it to add "realism," while still keeping the computations manageable.

If we restrict ourselves to simple, symmetrical objects, we can greatly simplify the problem of calculating surface brightness. This is not a severe restriction in itself, as you can break most "complex" objects down into combinations of simple elemental shapes. We will have to limit ourselves to "normal" (that is, head-on) views of the objects, because once we allow objects to be rotated, they lose most of their symmetry. Also, calculation time (particularly that for hidden surface removal) increases enormously.

We use the Lambertian (diffuse reflector) model to determine the surface brightness. We assume that light comes from a single, specific direction and then find the cosine of the angle between this reference vector and a surface normal vector. This gives us shade values from 1, where the surface faces the light source, to 0 at the terminator, where light just grazes the surface; and negative values where the surface is turned away from the light. Here is our first option: We can clip the brightness values at 0 so that areas facing away from the light have zero brightness, as an object lit by a single source (in deep space) would appear. Or, we can use the absolute value of the cosine so that the object appears to be lit by two identical sources on opposite sides. Either way is simple, so we allow for both cases by using a flag to determine what kind of illumination scheme we want to use for a whole scene or any individual shape making up the scene.

Although this simple model is valid for diffusely reflecting surfaces, it does not account for any contributions from specular surface reflections, secondary light sources (other

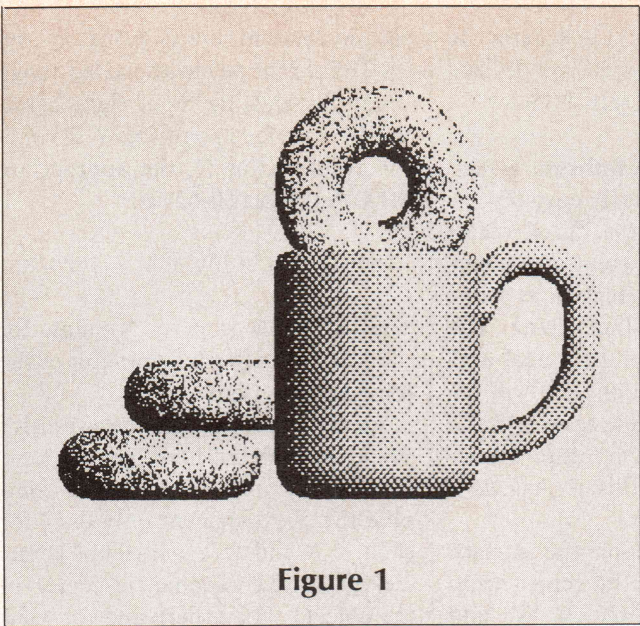


Figure 1

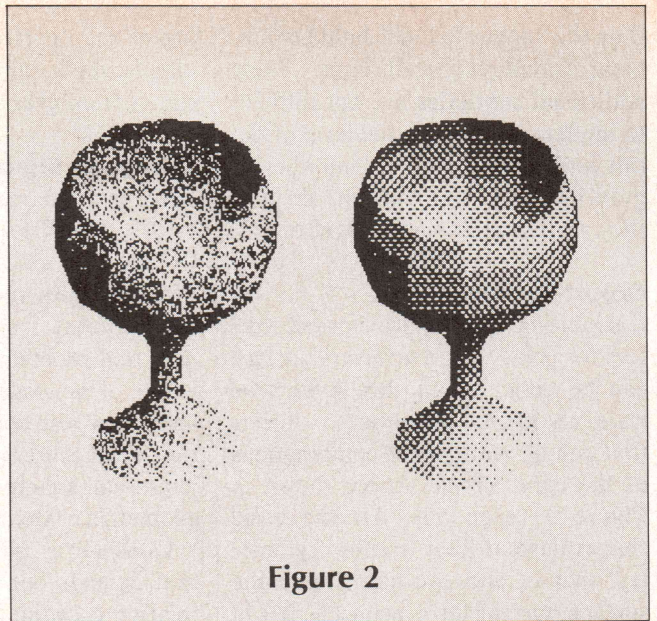


Figure 2

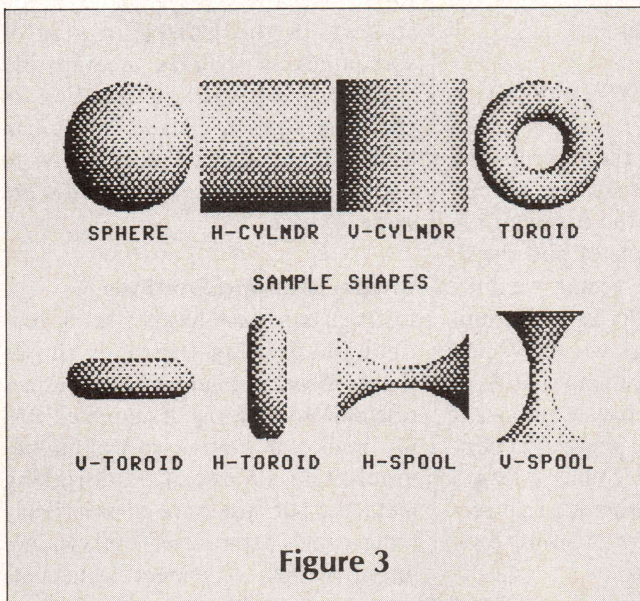


Figure 3

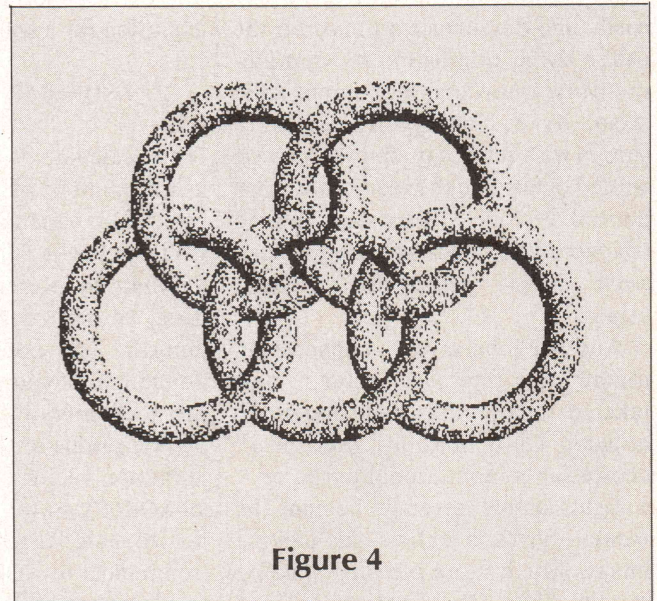


Figure 4

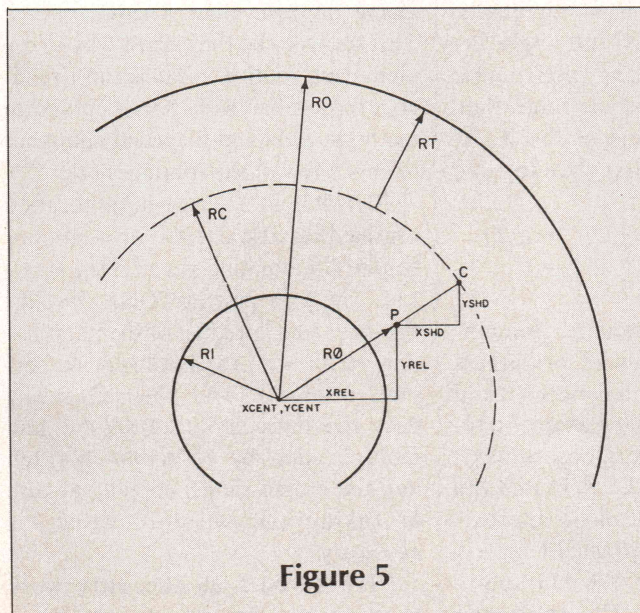


Figure 5

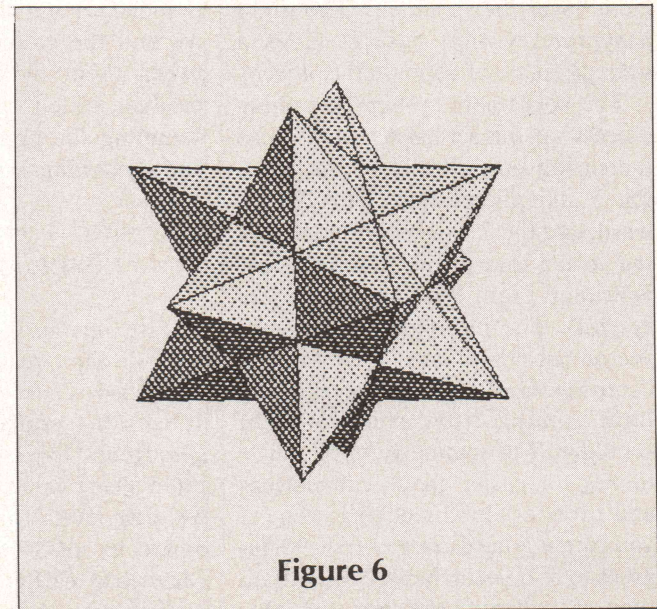


Figure 6

than the “opposite face” light) or ambient (nondirectional) light. These additional subtleties are not difficult to implement, but for the level of detail and dynamic range (number of gray levels) in our halftone display, they really aren’t worth the effort.

Object Geometry

Calculating the surface normal vectors for points on an arbitrary surface can be difficult, but this is why we have restricted ourselves to objects that can be made up of combinations of the eight simple shapes shown in Figure 3 (page 55). All the basic shapes have at least fourfold symmetry; that is, once we have found the surface normal for a point (X, Y) in one quadrant of the object, we can easily find the surface normals for the other three quadrants by complementing appropriate components. Also, by restricting ourselves to shapes with relatively simple surfaces, we can determine the required surface normal vectors without resorting to trigonometric functions, using instead some simple geometric considerations.

Another time-saver is to choose an illumination reference vector to best take advantage of symmetries in the objects. Unfortunately, the “best” choice for computational purposes—a light source directly behind the viewer—gives a rather flat-looking image. Much more pleasing shading results from illumination coming from “over the shoulder.” The slight assymetry that results gives a stronger sense of depth to the objects.

The coordinate system describing objects on the screen has its X axis increasing horizontally to the right, the Y axis increasing vertically upward, and the Z axis increasing out of the screen toward the viewer (a conventional right-handed coordinate system). The illumination reference vector used here has X, Y, and Z components of (1,1,2), representing light coming from over the right shoulder. The symmetry in X and Y makes for easier shade calculations and gives a similar looking scene if you rotate a hard-copy output 90 degrees to fit objects that are taller than they are wide into the display. Light-

ing in the “portrait” (as opposed to “landscape”—unrotated) mode then comes from over the left shoulder.

Drawing a Shaded Sphere

Spheres are particularly easy objects to deal with because a radial vector from the center to a point on the surface is in the same direction as a surface normal from that point. The only real problem we must deal with is one of normalization. We will *always* deal with coordinates relative to the “local center of curvature” of objects. For spheres, this is just their geometric center.

Consider a point on the surface that extends out of the screen from relative coordinates (X,Y). We find the corresponding Z value using the equation for a sphere:

$$Z = \text{SQR}(R^2 - X^2 - Y^2)$$

This calculated Z then forms the third member of the surface normal vector. To calculate the cosine of the angle between this vector and the illumination reference vector, we must first normalize both to have unit length. The normal we have just found has, of course, a length of R, so we need only divide each component by the radius of the sphere. The illumination vector (1,1,2) has a length of $\text{SQR}(6)$, so we normalize this vector to unit length by dividing each component by $\text{SQR}(6)$.

Now that we have two vectors of unit length but different directions, we find the cosine of the angle between them by taking their inner product, which is nothing more than summing the products of the X, Y, and Z coordinate pairs—like so:

$$\text{COSINE} = (1*X + 1*Y + 2*Z) / (R*\text{SQR}(6))$$

To use integer arithmetic throughout, we don’t really want brightness expressed as a fraction between 0 and 1, but rather scaled to a range of integers from 0 to 63. We choose a maximum shade value of 63 to match the 64 gray levels that can be approximated by an 8×8 threshold matrix (described earlier in the “Graphics Utilities” section). We scale the

pseudo-random bytes (used in the RSHADE routine) to the same range simply by shifting them right twice. So, for a surface point (X, Y, Z) on a sphere of radius R, the appropriate SHADE integer (0-63) is:

$$\text{SHADE} = 26*(X + Y + 2*Z)/R$$

The factor 26 properly accounts for the $\text{SQR}(6)$ in the denominator of the previous equation.

As stated earlier, all the primitive shapes considered here have at least fourfold symmetry. We need compute the Z component only once for each $\pm X$ and $\pm Y$ quartet of points (relative to the geometric center of the object). The sphere and top-view toroid actually have n-fold symmetry, which the Cartesian grid of screen pixels reduces to eightfold symmetry. That is, in addition to changing signs of X and Y, we can exchange the X and Y coordinates to find another surface point with the same Z value.

Shape-Drawing Routines

Listing Four (page 68) is a set of routines for drawing the eight simple shapes shown in Figure 5. The routines resemble some form of compiled BASIC in that they use no special bit manipulations or unusual addressing methods, but just have appropriately ordered calls to the lower-level routines set up earlier. Comment statements precede each shape-drawing routine, giving an equivalent BASIC routine. This seems to be the easiest way to explain each routine because most readers are familiar with BASIC program methods. I thus give special comment to only a few of the routines here.

GETVAL is a shade-normalization routine that also checks for normal or backlit illumination via the flag BACKLIT. The byte pair at TONE contains the inner (dot) product of the illumination vector with the local surface normal vector. GETVAL then effectively does the division by $\text{SQR}(6)*R$ and multiplication by 63 to put VALUE into the proper range, clipping at zero or taking the absolute value, as necessary.

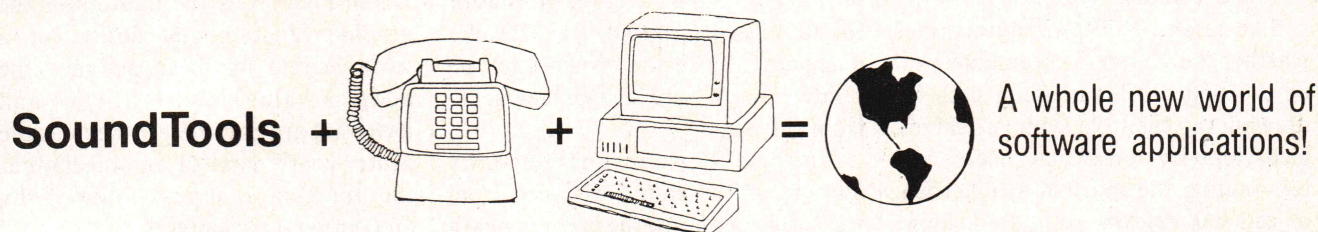
PTPLOT does all the dirty work needed to plot shaded points for four-

SOFTWARE DEVELOPERS

Add Touch Tone™ interface and Sound to your programs with

SoundTools™

Touch Tone data entry is now at your fingertips with this inexpensive programming package for the IBM PC, XT, AT and compatible personal computers.



Use Digital Pathways' SoundTools software (and Communicard™) to enhance your current applications and develop new ones that will:

- record and playback voice/sound
- provide remote data entry via Touch Tone
- perform automatic pulse and Touch Tone dialing
- detect call progress (ring, busy, voice, etc.)

SoundTools supports all of these languages... Interpreted and Compiled IBM BASICA and MS BASIC, MS PASCAL, C and Assembler.

List price only \$449

Includes SoundTools software, Communicard, modular telephone cord and user's manual.
Ask about 21 day trial offer when placing your order.

Communicard is a 1/2-size card that provides complete telephone interface, microphone and auxiliary output and Touch Tone decoding.

Requirements: PC/MS-DOS 2.0, 2.1, 3.0 or 3.1; IBM PC, XT, AT or compatible; 192K memory; DMA channel 1.

You too, can put your software on a SOUND FOOTING...

SoundWare™
S E R I E S O F S O F T W A R E

DIGITAL PATHWAYS, INC.

1060 EAST MEADOW CIRCLE, PALO ALTO, CA 94303, 415-493-5544

SoundTools, Communicard and SoundWare are registered trademarks of Digital Pathways, Inc.
IBM is a registered trademark of International Business Machines Corp.
MS is a registered trademark of Microsoft Corporation.
Touch Tone is a trademark of AT&T.



fold symmetrical objects. Provision is also made here for clipping the object at independent levels up, down, left, and right of the object's center. The clipping feature is needed to blend various primitive objects into more complex ones with smooth transitions at the seams. Clipping also allows us to create some intricate "weaving" effects by redrawing portions of overlapping shapes as in Figure 4, "Linked Toroids," (page 55).

The flag NOROT determines whether the X and Y coordinates will be exchanged. This is useful both for drawing the eightfold symmetrical objects (top-view toroid and sphere) and for rotating the fourfold symmetrical objects 90 degrees so that a single drawing routine can produce two orientations of an object. The calculations are basically those for the sphere, but we'll see later that we can put other objects, such as toroids, into forms that look like spheres (at least locally).

GETZ calculates the effective Z coordinate for a spherelike surface, given the X and Y coordinates relative to the local center of curvature. Note that this routine needs the SQRT function, and because it is in general called for each quartet of points plotted, our fast SQRT makes a big difference in execution speed.

SPHERE follows the previously outlined algorithm and takes full advantage of the available symmetry. The radius is POKed into RADIUS, and clipping distances relative to the sphere center are POKed into CLIPL (left), CLIPR (right), CLIPU (up), and CLIPD (down).

CYLNDR draws a cylinder with sizes POKed into RADIUS and HLEN (half-length). The flag NOROT determines whether the cylinder will be drawn with a horizontal or a vertical axis. The routine is simple, using PTPLOT for the main shade-calculation tasks. Actually, we could write a much faster routine to take advantage of the fact that we can use the same shade value for all points along lines parallel to the cylinder axis. This would require considerably more space, however.

The peculiar manipulations of relative coordinates for plotting toroids in various orientations are easier to

explain with the help of the diagram in Figure 5 (appropriate for a top-view toroid) (page 55).

Consider a point P at coordinates (XREL, YREL) relative to the center of the object. We can take the local center of curvature of the toroid to be at the point C, which lies on the intersection of the line passing through the center of the toroid and a point beneath P (where Z=0) with a circle of radius RC (the "average" radius of the toroid is $RC = (RO + RI) / 2$). We determine the surface normal by the relative X, Y, and Z displacements from the point C.

The byte R0 is used for temporary storage of the radial distance from the toroid center to the point beneath P (where Z=0):

$$R0 = \text{SQR}(XREL * XREL + YREL * YREL)$$

We can determine the relative X and Y displacements, XSHD and YSHD, respectively, of P from C easily by using similar triangles:

$$XSHD = XREL * (1 - RC / R0) \\ YSHD = YREL * (1 - RC / R0)$$

We find the Z coordinate via the Pythagorean theorem, using the radius of the "ring" portion, RT, as the hypotenuse. RT is then also the length of the normal vector and is the value POKed into RADIUS for normalization in the shade calculations.

The routine TOROID follows this algorithm, taking advantage of the eightfold symmetry in the object. EDGTOR and SPOOL use similar center of curvature coordinates to draw other orientations of a toroid. I have given the BASIC equivalents but leave it to readers to draw the appropriate geometric constructions if they desire more details.

Using the Shape-Drawing Routines—Interface to BASIC

The last module, completing our graphics package, is a convenient interface to BASIC. Listing Five (page 76) is a collection of routines to pass parameters easily to the graphics routines from the BASIC programs that control image generation. These rou-

tines are specific to the Commodore 64 in that they use some of the routines in the BASIC ROM to interpret statements in a BASIC program or entered in the direct command mode.

Graphics commands are given in the form:

SYS(FNCTN),PARAM1,PARAM2,
[OPTIONAL PARAMETERS]

where FNCTN is the address of the graphic function you desire (or a variable that has been assigned the address value) and PARAM1 and PARAM2 are parameters (such as the center coordinates of an object) that the function requires, followed by any optional parameters.

The parameters can be either literal numerics or expressions that are evaluated for the desired values. We do not have to put the function address in parentheses, but it helps to make the program more readable. Separating parameters by commas is required. If you desire an optional parameter (such as specifying a new radius for sphere drawing), you add it by following the last required parameter by a comma and then the optional value or expression.

The use of optional parameters is more easily explained with the sequence of BASIC statements found in Figure 8 (page 59):

For all the shape-drawing functions, you must specify the X and Y center coordinates, but shape sizes are optional parameters. The last values specified become the new default sizes, allowing you to draw copies of similar objects (such as a bunch of grapes) just by setting the new center coordinates. All shapes but the sphere take two size parameters, and you must specify both when you desire a change (even when only one parameter takes on a new value). All size parameters (radii and cylinder half-lengths) must be less than 256, not a severe limitation considering the screen is only 200 pixels high ("240" with scaling).

The commands to clear the bit map and initialize the color map each take a single optional parameter, but the default values remain unchanged. Unless you specify alternate bytes, the bit

map is filled with 0s (cleared) and the color map is filled with 1s (for black dots on a white background). The most useful alternate byte with which to fill the bit map is 255, setting the entire background. Because the shading process clears, as well as sets, appropriate points, objects still appear normal but against a black background (particularly effective when you use the "back light" style).

The byte used to fill the color map is made up of background and foreground nibbles. To initialize the color map for a different color combination than black on white, use:

```
SYS(52001),16*DC+BC
```

where DC is the dot-color number and BC is the background-color number (as given in any reference to programming the Commodore 64).

The addresses for all the graphic functions and the parameters they take, along with locations to be POKed with clipping values and flags for various drawing styles, are summarized in Table 3 (page 60). The best way to see how these shape-drawing routines are used is to examine the demonstration programs in Listing Six (page 78), "SHAPES DEMO," and Listing Seven (page 80), "STELLATION." All the different style options are exercised there. It is useful to save an abbreviated version of SHAPES DEMO, consisting of just the lines up to 340 and the sub-routines following line 1620. This skeleton program can then form a base, providing all the POKE and SYS locations you need and to which you can add your own image-generation control program.

Auxiliary Programs

SHAPES DEMO and STELLATION make use of two auxiliary programs that, while not strictly needed to produce graphic images, provide utilities to both enhance the image itself and speed the drawing process. Listing Eight (page 80) is a routine for sorting an integer array indirectly through a key array (to determine drawing order for facets in a polygon mesh quickly). Listing Nine (page 82) allows you to add text to graphic

```
10 REM ORDERED DITHER ARRAY—RECURSIVE FILL
20 I=0:J=0:K=1:N=0:P=3
30 REM 'P' IS ORDER OF ARRAY, SIZE IS (2 ^ P) x (2 ^ P)
40 P=2 ^ P:DIM A(P-1,P-1)
50 GOSUB 100
60 END
100 IF K=P THEN A(I,J)=N:N=N+1:K=K/2:RETURN
110 K=2*K:GOSUB 100
120 I=I+K:J=J+K:K=2*K:GOSUB 100
130 I=I-K:K=2*K:GOSUB 100
140 I=I+K:J=J-K:K=2*K:GOSUB 100
150 I=I-K:K=K/2:RETURN
```

Figure 7

```
... (SET UP GRAPHICS, STYLES, ETC)
200 SP=52119:REM ADDRESS OF SPHERE FUNCTION
210 SYS(SP),80,75,30:REM DRAW A SPHERE OF RADIUS 30 AT X=80, Y=75
220 SYS(SP),300,50:REM DRAW ANOTHER SPHERE AT X=300, y=50
230 REM SINCE NO RADIUS IS SPECIFIED, LAST VALUE IS USED AS DEFAULT
240 SYS(SP),200,150,40:REM NEW SPHERE RADIUS (40) BECOMES DEFAULT
```

Figure 8

FASTER C

TRY FASTER C RISK FREE
FOR 30 DAYS WITH
OUR MONEY BACK
GUARANTEE

helps Develop and Test Lattice C Programs

Reliably Cuts Both —

- Compile times (by 15% to 55%)
- Testing time (by 12% to 37%)

FASTER C keeps the Lattice C library and any other functions you choose in memory. It manages a jump table to replace the LINKER and immediately execute your functions. You can also CALL active functions interactively to speed your program debugging. It includes many options for configuration and control.

ONLY \$95.
CALL TOLL FREE
800-821-2492

for "Technical Description" or to order.

AVAILABLE FOR PC-DOS, IBM-AT,
AND ANY 256K MSDOS SYSTEM.

**Solution
Systems™**

335-D Washington St., Norwell, Mass. 02061
617-659-1571

FASTER C is a trademark of Solution Systems

Circle no. 102 on reader service card.

images in a variety of styles.

The code for these routines is short enough to be tucked in behind Commodore's DOS Wedge program so that, again, you don't lose any BASIC program space but maintain full compatibility with the wedge. If you are willing to give up the convenience of the wedge, however, you can easily relocate these routines to the end of the "interface" subprogram to keep

the graphics package in one piece.

The KEYSORT routine works with two 1-dimensional integer arrays of the same size. One is filled with some "priority" parameter (such as the "average Z" coordinates for the facets in a polygon mesh), and the other becomes a "key" array whose elements index those in the "priority" array in increasing order.

To use the routine, POKE the max-

imum element index of the arrays into location 140, the address of the first (0th) element of the key array into locations 251 (low byte) and 252 (high byte), and the base address of the priority array into 253 and 254. You can find these array base addresses easily because the Commodore 64 places the address of the "current BASIC variable" into locations 71 and 72. Setting the 0th ele-

BIT-MAP SCREEN : 40960-48959 (\$A000-\$BF3F)

COLOR MAP : 33792-34791 (\$8400-\$87E7)

CLIPPING BOUNDS (relative to object center):

893 (\$037D) : LEFT BOUND
894 (\$037E) : RIGHT BOUND
895 (\$037F) : UPPER BOUND
896 (\$0380) : LOWER BOUND

53280 (\$D020) : BORDER COLOR [0-15], POKE with 1 to match white screen

STYLE FLAGS:

838 (\$0346) : SHADE STYLE [0=random, 1=halftone]
839 (\$0347) : SCALE FLAG [0=normal (1:1), 1=scaled (3:4)]
868 (\$0364) : EDGES FLAG [0=normal, 1=add lines to facet edges]
871 (\$0367) : EDGE/LINE MODE [0=draw, 1=erase]
898 (\$0382) : LIGHTING STYLE [0=normal single source, 1=backlit]

FUNCTION LOCATIONS—CALL WITH "SYS(FNCTN), PARAMETERS." OPTIONAL PARAMETERS IN SQUARE BRACKETS, DEFAULT VALUES ARE USED IF NOT SPECIFIED

49378 (\$C0E2) : SWITCH TO GRAPHICS MODE
49411 (\$C103) : RETURN TO TEXT SCREEN

51979 (\$CB0B) : CLEAR[,CLEAR BYTE] FILL BIT MAP WITH CLEAR BYTE [DEFAULT = 0]
52001 (\$CB21) : COLOR[,COLOR BYTE] FILL COLOR MAP WITH COLOR BYTE

52023 (\$CB37) : PLOT,X,Y SET POINT AT (X,Y)
52026 (\$CB3A) : UNPLOT,X,Y CLEAR POINT AT (X,Y)

52049 (\$CB51) : LINE,X1,Y1,X2,Y2 DRAW A LINE BETWEEN (X1,Y1) AND (X2,Y2)

52052 (\$CB54) : FACET,X1,Y1,X2,Y2,X3,Y3,VA DRAW A SHADED TRIANGULAR FACET BETWEEN COORDINATE PAIRS (X1,Y1), (X2,Y2) AND (X3,Y3) USING "VA" SHADE VALUE (0:BLACK TO 64:WHITE)

CURVED SURFACES:

52119 (\$CB97) : SPHERE,X,Y[,R] DRAW A SPHERE CENTERED AT (X,Y) WITH RADIUS R
52141 (\$CBAD) : TOROID,X,Y[,RI,RO] DRAW A TOP-VIEW TOROID WITH INNER RADIUS RI AND OUTER RADIUS RO
52150 (\$CBB6) : VCYL,X,Y[,R,HL] DRAW A CYLINDER WITH AXIS VERTICAL, RADIUS R AND HALF-LENGTH HL
52153 (\$CBB9) : HCYL,X,Y[,R,HL] DRAW A CYLINDER WITH AXIS HORIZONTAL
52186 (\$CBDA) : VTOR,X,Y[,RI,RO] DRAW AN EDGE-VIEW TOROID WITH AXIS VERTICAL
52189 (\$CBDD) : HTOR,X,Y[,RI,RO] DRAW AN EDGE-VIEW TOROID WITH AXIS HORIZONTAL
52203 (\$CBEB) : VSPool,X,Y[,RI,RO] DRAW AN INSIDE-VIEW TOROID ("SPOOL") WITH AXIS VERTICAL
52206 (\$CBEE) : HSPool,X,Y[,RI,RO] DRAW AN INSIDE-VIEW TOROID ("SPOOL") WITH AXIS HORIZONTAL

Table 3
Graphics Addresses

ment of an array equal to itself merely establishes it as the current variable. Then you can transfer the contents of 71 and 72 to the proper pointer at 251/252 or 253/254. Be careful to use literal numerics for "251" and so on, because you don't want to change the current variable. After setting up the pointers, SYS to the start of the KEYSORT routine. Another caution to observe if you use KEYSORT for other applications is *not* to create any new variables between finding the array base addresses and invoking KEYSORT, because arrays will be pushed up in memory to keep them at the end of BASIC's variable storage.

KEYSORT is set up to handle arrays with up to 256 elements each. If needed, you can sort larger arrays by breaking them into smaller arrays, sorting with KEYSORT, and then collating the results. Merging a few already sorted arrays is a fairly simple operation that you can do in a single pass, extending the usefulness of this routine.

STELLATION uses KEYSORT (lines 840-880) to determine the drawing order for the facets of a small stellated dodecahedron (Figure 6, page 55). This is a polyhedron that is not strictly convex (some internal dihedral angles are greater than 180 degrees), causing some foreground facets to partially obscure those in the background. The Painter's algorithm is used to handle hidden surface removal. Each newly drawn fac-

et completely overwrites the background (clearing, as well as setting, points) so that drawing facets from back to front gives a proper rendering of the object.

The stellated dodecahedron of Figure 6 usually has only 30 visible or partially visible facets. Sorting with BASIC would be satisfactory here, but more complex polygon mesh objects can easily contain hundreds of visible and partially visible facets. You can really appreciate KEYSORT in those cases. Although it uses the simple and inefficient bubble-sort algorithm, being implemented in machine code lets KEYSORT run circles around any BASIC sorting alternative.

The final routine, in Listing Nine, WRITE, allows you to add text to the graphic display. The primary reason for implementing this as a machine code routine is that the bit map for the graphic image is located in the shadow RAM beneath the BASIC ROM (again saving useful BASIC program memory space). We can transfer bit images from the character ROM to the bit map just by POKEing values to the bit map, but we gain more flexibility if we first read the bit-map bytes to exclusive OR them, AND them, and so on with character image. Reading the shadow RAM requires that the BASIC ROM be switched out (through the bank switching used in the Commodore 64), something a BASIC program cannot do (and hope to contin-

ue execution).

The subroutine following line 1780 in SHAPES DEMO illustrates how to print text in a variety of styles on the bit-map image. Although the program actually uses only one style, the subroutine allows for five possibilities. Normal ("black" on "white") characters and reverse characters can overwrite the background, black characters can be ORed with the background, white characters ANDed with the background, or black characters exclusive ORed with the background. The last possibility gives characters that "change phase" across black/white edges in the image.

Conclusion

The graphics package described here we hope presents some new approaches to computer-generated images on small systems. At the same time, I'm sure I've missed some obvious tricks that would speed up the programs or reduce their size. I will be interested to see responses to this article and shortcuts or enhancements that might be added. Although the price/performance ratio of small graphics system continues to improve (especially with the development of custom LSI graphics chips), there are yet many unexplored software approaches to the problem. DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 194.

Drawing on the C-64 (Text begins on page 50)

Listing One

```
00001 0000 : INTEGER ARITHMETIC ROUTINES
00002 0000 : RICHARD L. RYLANDER 8/12/84
00003 0000 :
00004 0000 : REVISED 10/29/84 TO ADD FILL DOUBLE
00005 0000 : PRECISION ARGUMENTS IN DIVIDE ROUTINE
00006 0000 :
00007 0000 :
00008 0000 : *****
00009 0000 :
00010 0000 : USE PAGE ZERO LOCATIONS WHERE POSSIBLE FOR
00011 0000 : ITERATIVE PROCEDURE WORK SPACE
00012 0000 :
00013 0000 : MLCND =#AC : MULTIPLICAND
00014 0000 : MLFLR =#AD : MULTIPLIER
00015 0000 : PRD =#AE : PRODUCT
00016 0000 :
00017 0000 : DIVND =#FD : DIVIDEND/QUOTIENT
00018 0000 : DVSOR =#FE : DIVISOR
00019 0000 : RMNDR =#B4 : REMAINDER
00020 0000 :
00021 0000 : RADND =#AC : RADICAND
00022 0000 : ROOT =#B3C : SQUARE ROOT
00023 0000 :
00024 0000 : TEMP =#FB
00025 0000 :
00026 0000 : SET UP SEED VALUES FOR PSEUDO-RANDOM NUMBERS
00027 0000 : **ICW00
00028 0000 : RNDM .BYTE 1FF,#55
00029 0000 : RTCMP .BYTE 100,100
00030 0000 :
00031 0000 :
```

```
00032 0004 : *****
00033 0004 :
00034 0004 : MULTIPLY SINGLE PRECISION MULTIPLICAND
00035 0004 : BY SINGLE PRECISION MULTIPLIER GIVING
00036 0004 : DOUBLE PRECISION PRODUCT (ENTER AT "MULT")
00037 0004 :
00038 0004 : SPECIAL CASE: ENTER AT "SQUARE" TO FIND
00039 0004 : SQUARE OF SIGNED 8-BIT NUMBER
00040 0004 :
00041 0004 : A5 AC : SQUARE LDA MLCND : ENTRY TO SQUARE
00042 0006 : 10 07 : BPL POSITV : USE ABSOLUTE VALUE
00043 0008 : 38 : SEC : NEGATE IF NEEDED
00044 0009 : A9 00 : LDA #100
00045 000B : E5 AC : SBC MLCND
00046 000D : 85 AC : STA MLCND
00047 000F : 85 AD : POSITV STA MLFLR
00048 0011 : A9 00 : MULT LDA #100 : ENTRY TO MULTIPLY
00049 0013 : A2 08 : LDX #108
00050 0015 : 46 AD : LSR MLFLR
00051 0017 : 90 03 : BCC NOADD
00052 0019 : 18 : CLC
00053 001A : 65 AC : ADC MLCND
00054 001C : 6A : NOADD ROR A
00055 001D : 66 AE : ROR PRD
00056 001F : CA : DEX
00057 0020 : D0 F3 : BNE MLOOP
00058 0022 : 85 AF : STA PRD+1
00059 0024 : 60 : RTS
00060 0025 :
00061 0025 : *****
```

(Continued on next page)

Listing One

```

00062 C025      ;
00063 C025      ; DIVIDE DOUBLE PRECISION DIVIDEND
00064 C025      ; BY DOUBLE PRECISION DIVISOR GIVING
00065 C025      ; DOUBLE PRECISION QUOTIENT
00066 C025      ;
00067 C025      ; DIVIDEND IS REPLACED BY QUOTIENT
00068 C025      ; IN THE PROCESS
00069 C025      ;
00070 C025      ; QUOTIENT IS ROUNDED TO NEAREST INTEGER
00071 C025      ;
00072 C025 A9 00  DIVIDE LDA #00
00073 C027 B5 B4  STA RMNDR
00074 C029 B5 B4  STA RMNDR+1
00075 C02B A2 10  LDX #10
00076 C02D 26 FD  DLOOP ROL DVNDND
00077 C02F 26 FE  ROL DVNDND+1
00078 C031 26 B4  ROL RMNDR
00079 C033 26 B5  ROL RMNDR+1
00080 C035 38     SEC
00081 C03A A5 B4  LDA RMNDR
00082 C03B E5 B4  SBC DVNDND
00083 C03A A8     TAY
00084 C03B A5 B5  LDA RMNDR+1
00085 C03D E5 FC  SBC DVNDND+1
00086 C03F 90 04  BCC DECCNT
00087 C041 B4 B4  STY RMNDR
00088 C043 B5 B5  STA RMNDR+1
00089 C045 CA     DECCNT DEX
00090 C046 D0 E5  BNE DLOOP
00091 C048 26 FD  ROL DVNDND      ; CHECK IF REMAINDER
00092 C04A 26 FE  ROL DVNDND+1  ; IS >= 1/2 OF DIVIDEND
00093 C04C 06 B4  ASL RMNDR      ; FOR ROUNDING
00094 C04E 26 B5  ROL RMNDR+1
00095 C050 0B 0B  BCS ROUND
00096 C052 38     SEC
00097 C053 A5 B4  LDA DVNDND
00098 C055 E5 B4  SBC RMNDR
00099 C057 A5 FC  LDA DVNDND+1
00100 C059 E5 B5  SBC RMNDR+1
00101 C05B 0B 0B  BCS NOCHNG
00102 C05D E6 FD  ROUND INC DVNDND
00103 C05F D0 02  BNE NOCHNG
00104 C061 E6 FE  INC DVNDND+1
00105 C063 60     NOCHNG RTS
00106 C064      ;
00107 C064      ; *****
00108 C0A4      ;
00109 C064      ; TAKE INTEGER SQUARE ROOT OF A
00110 C064      ; DOUBLE PRECISION RADICAND GIVING
00111 C064      ; SINGLE PRECISION ROOT ( <= REAL ROOT )
00112 C064      ;
00113 C064 A2 0B  SORT LDX #0B
00114 C066 A9 00  LDA #00
00115 C068 8D 3C 03 STA ROOT
00116 C06B 8D 3C 03 STA ROOT+1
00117 C06E 85 FB  STA TEMP
00118 C070 85 FC  STA TEMP+1
00119 C072 0E 3C 03 SORT1 ASL ROOT
00120 C075 2E 3C 03 ROL ROOT+1
00121 C078 EE 3C 03 ROL ROOT
00122 C07B D0 03  BNE NEXT1      ; ASSUME CURRENT LSB OF
00123 C07D EE 3D 03 INC ROOT      ; ROOT WILL BE 1
00124 C080 06 AC  NEXT1 ASL RADICND
00125 C082 26 AD  ROL RADICND+1
00126 C084 26 FB  ROL TEMP
00127 C086 26 FC  ROL TEMP+1
00128 C088 06 AC  ASL RADICND
00129 C08A 26 AD  ROL RADICND+1
00130 C08C 26 FB  ROL TEMP
00131 C08E 26 FC  ROL TEMP+1
00132 C090 38     SEC
00133 C091 A5 FB  LDA TEMP
00134 C093 E0 3C 03 SBC ROOT
00135 C096 A8     TAY
00136 C097 A5 FC  LDA TEMP+1
00137 C099 E0 3D 03 SBC ROOT+1
00138 C09C 90 12  BCC RESTOR
00139 C09E 85 FC  STA TEMP
00140 C0A0 84 FB  STY TEMP
00141 C0A2 EE 3C 03 INC ROOT
00142 C0A5 D0 03  BNE NEXT2
00143 C0A7 EE 3D 03 INC ROOT+1
00144 C0AA CA     NEXT2 DEX
00145 C0AB D0 C5  BNE SORT1
00146 C0AD 4C C1 C0 JMP FINI
00147 C0B0 38     RESTOR SEC
00148 C0B1 AD 3C 03 LDA ROOT
00149 C0B4 E9 01  STA #01
00150 C0B6 8D 3C 03 SBC ROOT
00151 C0B9 80 03  BCS NEXT3
00152 C0BB CE 3D 03 DEC ROOT+1
00153 C0BE CA     NEXT3 DEX
00154 C0BF D0 B1  BNE SORT1
00155 C0C1 6E 3D 03 FINI ROR ROOT+1
00156 C0C4 6E 3C 03 ROR ROOT
00157 C0C7 60     RTS
00158 C0C8      ;
00159 C0C8      ; *****
00160 C0C8      ;
00161 C0C8      ; GENERATE PSEUDO-RANDOM BYTES
00162 C0C8      ; EXIT WITH P-R BYTE IN ACCUM.
00163 C0C8      ;
00164 C0C8 AD 00 C0 RANDOM LDA RNDM
00165 C0CB 8D 02 C0 STA RTEMP
00166 C0CE 4D 01 C0 EOR RNDM+1
00167 C0D1 2E 03 C0 ROL RTEMP+1
00168 C0D4 6A     ROR A
00169 C0D5 6E 03 C0 ROR RTEMP+1
00170 C0D8 8D 00 C0 STA RNDM
00171 C0DB AD 02 C0 LDA RTEMP
00172 C0DE 8D 01 C0 STA RNDM+1
00173 C0E1 60     RTS
00174 C0E2      ;.END

```

END OF ASSEMBLY

End Listing One

Listing Two

```

00001 0000      ;
00002 0000      ; GRAPHICS UTILITIES
00003 0000      ;
00004 0000      ; RICHARD L. RYLANDER 11/4/84
00005 0000      ;
00006 0000      ; LOAD ARITHMETIC UTILITIES FIRST
00007 0000      ;
00008 0000      ; RAM=#037E
00009 0000      ; ORIGIN=#C0E2
00010 0000      ;
00011 0000      ; MLCND=#AC      ; MULTIPLICAND (S)
00012 0000      ; MLPLER=#AD     ; MULTIPLIER (S)
00013 0000      ; PROD=#AE     ; PRODUCT (D)
00014 0000      ; MULT=#C011   ; CALL FOR MULTIPLY
00015 0000      ;
00016 0000      ; RNDM=#C000      ; RANDOM NUMBER
00017 0000      ; RANDOM=#C0CB      ; CALL FOR RANDOM
00018 0000      ; NOTE - A CALL TO 'RANDOM' LEAVES A RANDOM BYTE
00019 0000      ; IN THE ACCUMULATOR
00020 0000      ;
00021 033E      ; **RAM
00022 033F      ;
00023 0341      ; PLTFLG ***+1      ; PLOT/UNPLOT FLAG
00024 0342      ; XFLT ***+2      ; ABSOLUTE PLOT COORD
00025 0343      ; YFLT ***+1      ; ABSOLUTE PLOT COORD
00026 0344      ; VIC1 ***+1      ; REGISTER STORAGE
00027 0346      ; VIC2 ***+1      ; REGISTER STORAGE
00028 0347      ; VALUE ***+2      ; FINAL NORMALIZED SHADE VALUE
00029 0348      ; HTOFRN ***+1      ; SHADE FLAG, 1=HALFTONE
00030 034A      ; NOSCALE ***+1      ; SCALE FLAG, 1=NO SCALE
00031 034A      ; TEMP ***+2      ; TEMPORARY STORAGE
00032 034A      ;
00033 034A      ; **ORIGIN
00034 034A      ;
00035 034A      ; *****
00036 034A      ;
00037 034A      ; TURN ON BIT MAP GRAPHICS MODE,
00038 034A      ; SAVING REGISTER VALUES FOR
00039 034A      ; RETURN TO TEXT MODE LATER.
00040 034A      ;
00041 034A      ;
00042 034A      ; GRFOFF LDA #D011
00043 034A      ; ORA #A20
00044 034A      ; STA #D011
00045 034A      ; LDA #D000
00046 034A      ; STA VIC1
00047 034A      ; AND #FC
00048 034A      ; ORA #01
00049 034A      ; STA #D000
00050 034A      ; LDA #D010
00051 034A      ; STA VIC2
00052 034A      ; LDA #19
00053 034A      ; STA #D018
00054 034A      ; RTS
00055 034A      ;
00056 034A      ; *****
00057 034A      ;
00058 034A      ; RETURN TO TEXT SCREEN
00059 034A      ;
00060 034A      ;
00061 034A      ; GRFOFF LDA #D011
00062 034A      ; AND #DF
00063 034A      ; STA #D011
00064 034A      ; LDA VIC1
00065 034A      ; STA #D000
00066 034A      ; LDA VIC2
00067 034A      ; STA #D018
00068 034A      ; RTS
00069 034A      ;
00070 034A      ; *****
00071 034A      ;
00072 034A      ; FILL COLOR MAP FOR BLACK DOTS ON WHITE
00073 034A      ;
00074 034A      ;
00075 034A      ; COLOR LDA #01      ; POKE NEW COLORS HERE
00076 034A      ; LDX #0
00077 034A      ; COL1 STA #B100,X
00078 034A      ; STA #B500,X
00079 034A      ; STA #B600,X
00080 034A      ; STA #B700,X
00081 034A      ; DEX
00082 034A      ; BNE COL1
00083 034A      ; RTS
00084 034A      ;
00085 034A      ; *****
00086 034A      ;
00087 034A      ; CLEAR HI-RES GRAPHICS SCREEN
00088 034A      ;
00089 034A      ;
00090 034A      ; CLEAR LDA #A0
00091 034A      ; STA #FC
00092 034A      ; LDY #0
00093 034A      ; STY #FB
00094 034A      ; LDA #0
00095 034A      ; LDX #20
00096 034A      ; CLRLP STA (#FB),Y
00097 034A      ; INY
00098 034A      ; BNE CLRLP
00099 034A      ; INC #FC
00100 034A      ; DEX
00101 034A      ; BNE CLRLP
00102 034A      ; RTS
00103 034A      ;
00104 034A      ; *****
00105 034A      ;
00106 034A      ; PLOT AND UNPLOT POINTS ON HI-RES GRAPHICS
00107 034A      ; SCREEN. ABSOLUTE X AND Y SCREEN COORDINATES
00108 034A      ; ARE POKED INTO XFLT, XFLT+1, AND YFLT
00109 034A      ;
00110 034A      ;
00111 034A      ; PLOT LDA #0
00112 034A      ; .BYTE #2C
00113 034A      ; UNPLOT LDA #80
00114 034A      ; STA PLTFLG
00115 034A      ; LDA #01
00116 034A      ; STA #FB
00117 034A      ; AND #FE
00118 034A      ; STA #01
00119 034A      ; SEC
00120 034A      ; LDA #C7
00121 034A      ; SBC YFLT
00122 034A      ; TAX
00123 034A      ; LSR A
00124 034A      ; LSR A
00125 034A      ; LSR A
00126 034A      ; TAY
00127 034A      ; LDA TABLE1,Y
00128 034A      ; STA #FB
00129 034A      ; LDA TABLE2,Y
00130 034A      ; STA #FC

```

(Continued on page 64)

DIGITAL RESEARCH COMPUTERS

(214) 225-2309

DISK DRIVE SPECIAL!

5 1/4" Double Sided-Double Density

Brand New, Unused. Mfg. in Japan by Canon. 2/3 Height - Direct Drive! Industry Standard Pin Out. 6MS Access. 40 Tracks.

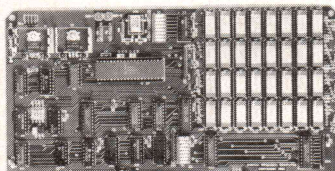
\$49⁹⁵
ea

2 FOR \$85

Sold on a First Come Basis
Add \$2 Each U.P.S.

256K S-100 SOLID STATE DISK SIMULATOR!
WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.

PRICE CUT!



BLANK PCB
(WITH CP/M* 2.2
PATCHES AND INSTALL
PROGRAM ON DISKETTE)
\$69⁹⁵
(8203-1 INTEL \$29.95)

- FEATURES:
- * 256K on board, using + 5V 64K DRAMS.
 - * Uses new Intel 8203-1 LSI Memory Controller.
 - * Requires only 4 Dip Switch Selectable I/O Ports.
 - * Runs on 8080 or Z80 S100 machines.
 - * Up to 8 LS-100 boards can be run together for 2 Meg. of On Line Solid State Disk Storage.
 - * Provisions for Battery back-up.
 - * Software to mate the LS-100 to your CP/M* 2.2 DOS is supplied.
 - * The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
 - * Compare our price! You could pay up to 3 times as much for similar boards.

\$199⁰⁰

#LS-100 (FULL 256K KIT)

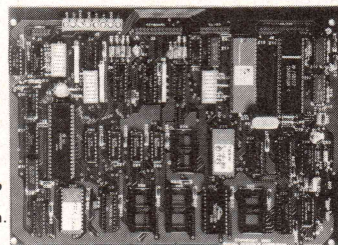
THE NEW ZRT-80

CRT TERMINAL BOARD!

A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.

FEATURES:

- * Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
- * RS232 at 16 BAUD Rates from 75 to 19,200.
- * 24 x 80 standard format (60 Hz).
- * Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
- * Higher density formats require up to 3 additional 2K x 8 6116 RAMS.
- * Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
- * 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
- * Composite or Split Video.
- * Any polarity of video or sync.
- * Inverse Video Capability.
- * Small Size: 6.5 x 9 inches.
- * Upper & lower case with descenders.
- * 7 x 9 Character Matrix.
- * Requires Par. ASCII keyboard.



BLANK PCB WITH 2716
CHAR. ROM, 2732 MON. ROM
\$49⁹⁵

SOURCE DISKETTE - ADD \$10
SET OF 2 CRYSTALS - ADD \$7.50

\$99⁹⁵

ZRT-80

(COMPLETE KIT,
2K VIDEO RAM)

WITH 8 IN.
SOURCE DISK!
(CP/M COMPATIBLE)

64K S100 STATIC RAM

\$139⁰⁰
KIT

NEW!

LOW POWER!
150 NS ADD \$10

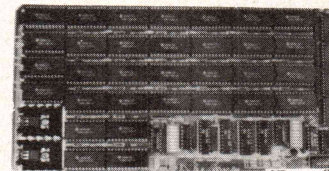
BLANK PC BOARD
WITH DOCUMENTATION
\$49.95

SUPPORT ICs + CAPS
\$17.50

FULL SOCKET SET
\$14.50

FULLY SUPPORTS THE
NEW IEEE 696 S100
STANDARD
(AS PROPOSED)

FOR 56K KIT \$125
ASSEMBLED AND
TESTED ADD \$50



FEATURES: **PRICE CUT!**

- * Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- * Fully supports IEEE 696 24 BIT Extended Addressing.
- * 64K draws only approximately 500 MA.
- * 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- * SUPPORTS PHANTOM (BOTH LOWER 32K AND ENTIRE BOARD).
- * 2716 EPROMs may be installed in any of top 48K.
- * Any of the top 8K (E000 H AND ABOVE) may be disabled to provide windows to eliminate any possible conflicts with your system monitor, disk controller, etc.
- * Perfect for small systems since BOTH RAM and EPROM may co-exist on the same board.
- * BOARD may be partially populated as 56K.

64K SS-50 STATIC RAM

\$119⁰⁰
(48K KIT)

NEW!

LOW POWER!
RAM OR EPROM!

BLANK PC BOARD
WITH
DOCUMENTATION
\$52

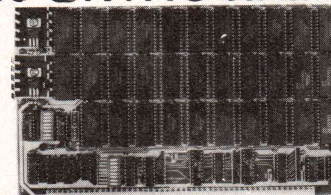
SUPPORT ICs + CAPS
\$18.00

FULL SOCKET SET
\$15.00

56K Kit \$129

64K Kit \$139

ASSEMBLED AND
TESTED ADD \$50



FEATURES:

- * Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- * Fully supports Extended Addressing.
- * 64K draws only approximately 500 MA.
- * 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- * Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- * 2716 EPROMs may be installed anywhere on Board.
- * Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- * One Board supports both RAM and EPROM.
- * RAM supports 2MHZ operation at no extra charge!
- * Board may be partially populated in 16K increments.

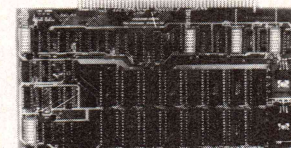
32K S100 EPROM/STATIC RAM

NEW!

FOUR FUNCTION BOARD!

NEW!

EPROM II
FULL
EPROM KIT
\$69.95
A&T EPROM
ADD \$35.00



BLANK
PC BOARD
WITH DATA
\$39.95

SUPPORT
IC'S
PLUS CAPS
\$16

FULL
SOCKET SET
\$15

We took our very popular 32K S100 EPROM Card and added additional logic to create a more versatile EPROM/RAM Board.

FEATURES:

- * This one board can be used in any one of four ways:
A. As a 32K 2716 EPROM Board
B. As a 32K 2732 EPROM Board (Using Every Other Socket)
C. As a mixed 32K 2716 EPROM/2K x 8 RAM Board
D. As a 32K Static RAM Board
- * Uses New 2K x 8 (TMM2016 or HM6116) RAM's
- * Fully supports IEEE 696 Buss Standard (As Proposed)
- * Supports 24 Bit Extended Addressing
- * 200 NS (FAST!) RAM's are standard on the RAM Kit
- * Supports both Cromemco and North Star Bank Select
- * Supports Phantom
- * On Board wait State Generator
- * Every 2K Block may be disabled
- * Addressed as two separate 16K Blocks on any 64K Boundary
- * Perfect for MP/M* Systems
- * RAM Kit is very low power (300 MA typical)

32K STATIC RAM KIT — \$109.95
For RAM Kit A&T — Add \$40

TERMS: Add \$3.00 postage. We pay balance. Orders under \$15 add 75¢ handling. No. C.O.D. We accept Visa and MasterCard. Tex. Res. add 5-1/8% Tax. Foreign orders (except Canada) add 20% P & H. Orders over \$50, add 85¢ for insurance.

PRICES
SLASHED!

Digital Research Computers

P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 225-2309

Drawing on the C-64

(Listing Continued, text begins on page 50)

Listing Two

```

00123 C166 0A      TXA
00124 C167 29 07   AND #007
00125 C169 18      CLC
00126 C16A 65 FB   ADC #FB
00127 C16C 95 FB   STA #FB
00128 C16E AD 3F 03 LDA XPLT
00129 C171 29 FB   AND #FB
00130 C173 65 FB   ADC #FB
00131 C175 65 FB   STA #FB
00132 C177 AD 40 03 LDA XPLT+1
00133 C17A 65 FC   ADC #FC
00134 C17C 95 FC   STA #FC
00135 C17E A9 00   LDA #00
00136 C180 65 FC   ADC #FC
00137 C182 85 FC   STA #FC
00138 C184 AD 3F 03 LDA XPLT
00139 C187 29 07   AND #007
00140 C189 47 07   EOR #007
00141 C18B AA      TAX
00142 C18C A9 01   LDA #01
00143 C18E CA      FLOTLP DEX
00144 C18F 30 03   BMI PLOT2
00145 C191 0A      ASL A
00146 C192 D0 FA   BNE PLOTLP
00147 C194 A0 00   LDY #0
00148 C196 2C 3E 03 BIT PLTFLG
00149 C199 10 05   BPL NOFLOT
00150 C19B 49 FF   EOR #FF
00151 C19D 31 FB   AND (#FB),Y
00152 C19F 2C      .BYTE #2C
00153 C1A0 11 FB   NOFLOT ORA (#FB),Y
00154 C1A2 91 FB   STA (#FB),Y
00155 C1A4 A5 01   LDA #01
00156 C1A6 09 01   ORA #01
00157 C1A8 85 01   STA #01
00158 C1AA 60      RTS
00159 C1AB
00160 C1AB 00      TABLE1 .BYTE #00,#40,#80,#C0
00160 C1AC 40
00160 C1AD 00
00160 C1AE C0
00161 C1AF 00
00161 C1B0 40
00161 C1B1 80
00161 C1B2 C0
00162 C1B3 00
00162 C1B4 40
00162 C1B5 80
00162 C1B6 C0
00163 C1B7 00
00163 C1B8 40
00163 C1B9 80
00163 C1BA C0
00164 C1BB 00
00164 C1BC 40
00164 C1BD 80
00164 C1BE C0
00165 C1BF 00
00165 C1C0 40
00165 C1C1 80
00165 C1C2 C0
00165 C1C3 00
00166 C1C4
00167 C1C4 00      TABLE2 .BYTE #00,#01,#02,#03
00167 C1C5 01
00167 C1C6 02
00167 C1C7 03
00168 C1C8 05
00168 C1C9 06
00168 C1CA 07
00168 C1CB 08
00169 C1CC 0A
00169 C1CD 0B
00169 C1CE 0C
00169 C1CF 0D
00170 C1D0 0F
00170 C1D1 10
00170 C1D2 11
00170 C1D3 12
00171 C1D4 14
00171 C1D5 15
00171 C1D6 16
00171 C1D7 17
00172 C1D8 19
00172 C1D9 1A
00172 C1DA 1B
00172 C1DB 1C
00172 C1DC 1E
00173 C1DD
00174 C1DE
00175 C1DF
00176 C1E0
00177 C1E1
00178 C1E2 AD 3F 03 SHADE LDA XPLT
00179 C1E3 29 07   AND #007
00180 C1E4 80 4B 03 STA TEMP
00181 C1E5 AD 41 03 LDA YPLT
00182 C1E6 29 07   AND #007
00183 C1E8 0A      ASL A
00184 C1E9 0A      ASL A
00185 C1EA 0A      ASL A
00186 C1EB 0D 4B 03 ORA TEMP
00187 C1EC C1F0 A0 2F C2 LDA THRESH,X
00188 C1F1 80 2F C2 CMP VALUE
00189 C1F2 CD 44 03 BPL MORE
00190 C1F3 10 03   BPL GREATR
00191 C1F4 4C 46 C1 JMP UNFLOT
00192 C1F5 4C 43 C1 JMP PLOT
00193 C1F6
00194 C1F7
00195 C1F8
00196 C1F9
00197 C1FA
00198 C1FB 20 C8 C0 RSHADE JSR RANDOM
00199 C200 2A 4A      LSR A
00200 C201 4A      CMP VALUE
00201 C202 10 03   BPL MORE
00202 C203 CD 44 03 BPL MORE
00203 C204 4C 46 C1 JMP UNFLOT
00204 C205 C206 4C 43 C1 JMP PLOT
00205 C207
00206 C208
00207 C209
00208 C20A
00209 C20B
00210 C20C
00211 C20D
00212 C20E
00213 C20F AD 47 03 PLTSHD LDA NOSCAL

```

```

00214 C212 F0 10   BEQ NORM
00215 C214
00216 C214
00217 C214
00218 C214
00219 C214
00220 C214 AC 41 03 SCALE LDY YPLT
00221 C217 C8      INY
00222 C218 84 AD   STY MLPLER
00223 C21A A9 D5   LDA #D5
00224 C21C 85 AC   STA MLFCND
00225 C21E 20 11 C0 JSR MULT
00226 C221 8D 41 03 STA YPLT
00227 C222 AD 46 03 LDA HTDRRN
00228 C227 F0 03   BEQ RFLT
00229 C229 4C DD C1 JMP SHADE
00230 C22C 4C FF C1 RFLT JMP RSHADE
00231 C22F
00232 C22F
00233 C22F
00234 C22F 00
00234 C230 00
00234 C231 35
00234 C232 30
00235 C233 02
00235 C234 0A
00235 C235 37
00235 C236 3F
00236 C237 10
00236 C238 18
00236 C239 25
00236 C23A 2D
00237 C23B 12
00237 C23C 1A
00237 C23D 27
00237 C23E 2F
00238 C23F 31
00238 C240 39
00238 C241 04
00238 C242 0C
00239 C243 33
00239 C244 3B
00239 C245 06
00239 C246 0E
00240 C247 21
00240 C248 29
00240 C249 14
00240 C24A 1C
00241 C24B 23
00241 C24C 2B
00241 C24D 16
00241 C24E 1E
00242 C24F 03
00242 C250 0B
00242 C251 36
00242 C252 3E
00243 C253 01
00243 C254 09
00243 C255 34
00243 C256 3C
00244 C257 13
00244 C258 1B
00244 C259 26
00244 C25A 2E
00245 C25B 11
00245 C25C 19
00245 C25D 24
00245 C25E 2C
00246 C25F 32
00246 C260 3A
00246 C261 0F
00246 C262 07
00247 C263 30
00247 C264 3B
00247 C265 05
00247 C266 0D
00248 C267 22
00248 C268 2A
00248 C269 17
00248 C26A 1F
00249 C26B 20
00249 C26C 2B
00249 C26D 15
00249 C26E 1D
00250 C26F

```

ERRORS = 00000

SYMBOL TABLE

SYMBOL VALUE

SYMBOL	VALUE	SYMBOL	VALUE	SYMBOL	VALUE	SYMBOL	VALUE
CLEAR	C12C	CLRLP	C13B	COL1	C11C	COLOR	C11B
GREATR	C1FC	GRFOFF	C103	GRFON	C0E2	HTDRRN	0346
MLPCND	00AC	MLPLER	00AD	MORE	C20C	MULT	C011
NOFLOT	C1A0	NORM	C224	NOSCAL	0347	ORIGIN	C0E2
PLT	C143	PLOT2	C194	PLTFLG	C18E	PLTFLG	033E
PLTSHD	C20F	PROD	00AE	RAM	033E	RANDOM	C0CB
RNDM	C000	RPLT	C22C	RSHADE	C1FF	SCALE	C214
SHADE	C1DD	TABLE1	C1AB	TABLE2	C1C4	TEMP	034B
THRESH	C22F	UNFLOT	C146	VALUE	0344	VIC1	0342
VIC2	0343	XPLT	033F	YPLT	0341		

END OF ASSEMBLY

End Listing Two

Listing Three

```

00001 0000
00002 0000
00003 0000
00004 0000
00005 0000
00006 0000
00007 0000
00008 0000
00009 0000
00010 0000
00011 0000
00012 0000
00013 0000
00014 0000
00015 0000
00016 0000
00017 0000

```

; FACET - DRAW SHADED TRIANGULAR FACETS
 ; AND STRAIGHT LINES.
 ;
 ; RICHARD L. RYLANDER 11/4/84
 ;
 ; LOAD "ARITH.HEX" AND "GRAPH.HEX"
 ; BEFORE USING
 ;
 ORIGIN = #C26F
 RAM = #034A
 ;
 XPLT = #033F
 YPLT = #0341
 ;
 NORM = #C224
 NOSCAL = #0347
 PLOT = #C143
 UNFLOT = #C146

(Continued on page 66)

Starlight FORTH for
65SC802 & 65SC816
 The new 16 bit CMOS 6502's from
 Western Design Center and GTE

- * Apple
- * Atari
- * AIM 65
- * Commodore
- * Ohio Scientific

- * Highly Extended fig-FORTH
- * Full Double Number Set - Strings
- * Optional Tools: Meta-Cross-Compiler with Assembler and Debugger
- * Multi-Tasking Super Fast FORTH-83 soon

Your present 8 bit system will run faster
 with 16 bit software (FORTH) and a 65SC802
 and still runs your existing 8 bit software

Starlight FORTH Systems

15247 N. 35th St.
 Phoenix, AZ 85032
 (602) 992-5181

Circle no. 60 on reader service card.

SMALL C FOR IBM-PC

Small-C Compiler Version
 2.1 for PC-DOS/MS-DOS
 Source Code included
 for Compiler & Library
 New 8086 optimizations
 Rich I/O & Standard Library

\$40

CBUG SOURCE LEVEL DEBUGGER FOR SMALL C

Break, Trace, and Change
 variables all on the
 source level
 Source code included

\$40

Datalight

11557 8th Ave. N.E.
 Seattle, Washington 98125
 (206) 367-1803

ASM or MASM is required with compiler.
 Include disk size (160k/320k), and DOS version with order.
 VISA & MasterCard accepted. Include card no. & expiration date.
 Washington state residents include 7.9% sales tax.
 IBM-PC & PC-DOS are trademarks of International Business Machines
 MS-DOS is a trademark of Microsoft Corporation.

Circle no. 6 on reader service card.

DATESTAMPER™ has the answers

Drive B1: 4 files, using 15K 110K FREE 14:01-03 Feb					
-- file	size	created	accessed	modified	
B1: ADDRESS .DAT	5K	22:01-17 Jan	08:30-01 Feb	08:23-01 Feb	
B1: JSMITH .LTR	2K	16:30-24 Dec '84	11:59-10 Feb	16:30-24 Dec '84	
B1: TEST1 .BAS	4K	09:34-22 Jan	16:27-30 Jan	09:35-22 Jan	
B1: TEST2 .BAS	4K	11:55-01 Feb		11:55-01 Feb	

When did we
print that letter?

Has the mailing
list been updated?

Which is the
latest version?

DateStamper™ keeps your CP/M computer up-to-date!

- avoid erasing the wrong file
- keep dated tax records of computer use
- back-up files by date and time
- simplify disk housekeeping chores

OPERATION: DateStamper extends CP/M 2.2 to automatically record the date and time a file is created, read or modified. DateStamper reads the exact time from the real-time clock, if you have one; otherwise, it records the order in which you use files. Disks initialized for datestamping are fully compatible with standard CP/M.

INSTALLATION: Default (relative-clock) mode is automatic. Configurable for any real-time clock, with pre-assembled code supplied for popular models. Loads automatically at power-on. **UTILITIES:** • Enhanced SuperDirectory • Powerful, all-function DATSWEEP file-management program with date and time tagging • Disk-initializer • Installation and configuration utilities

PERFORMANCE: Automatic. Efficient. Invisible. Compatible.

Requires CP/M 2.2. Uses less than 1K memory. Real-time clock is optional.

When ordering please specify format

8" SSSD, Kaypro, Osborne Formats \$49

For other formats (sorry, no 96 TPI) add \$5.

Shipping and handling \$3

California residents add 6% sales tax

MasterCard and Visa accepted

Specialized versions of this and other software available for the Kaypro.
 CP/M is a registered trademark of Digital Research, Inc.

Write or call for further information

Plu*Perfect Systems

BOX 1494 • IDYLLWILD, CA 92349 • 714-659-4432

Circle no. 69 on reader service card.

Drawing on the C-64

(Listing Continued, text begins on page 50)

Listing Three

```

00018 0000      ;
00019 0000      ; MLPCND = #AC
00020 0000      ; MLPLER = #AD
00021 0000      ; FRDD = #AE
00022 0000      ; MULT = #C011
00023 0000      ;
00024 0000      ; DVND = #FD
00025 0000      ; DVSOR = #FB
00026 0000      ; QUOT = #FD
00027 0000      ; DIVIDE = #C025
00028 0000      ;
00029 0000      ; **RAM
00030 034A      ;
00031 034A      ; XMIN ***+2
00032 034C      ; YMIN ***+2
00033 034D      ; XMID ***+2
00034 034F      ; YMID ***+2
00035 0350      ; XMAX ***+2
00036 0352      ; YMAX ***+1
00037 0353      ; YTOP ***+1
00038 0354      ; YBOT ***+1
00039 0355      ; YBASE ***+1
00040 0356      ; DLTA1 ***+2
00041 0358      ; DLTA2 ***+1
00042 0359      ; DLTA3 ***+1
00043 035A      ; DELTA ***+1
00044 035B      ; DLTA1 ***+1
00045 035C      ; DLTA2 ***+1
00046 035D      ; DLTA3 ***+1
00047 035E      ; DELTA ***+1
00048 035F      ; XDIFF ***+1
00049 0360      ; FLAG1 ***+1
00050 0361      ; FLAG2 ***+1
00051 0362      ; FLAG3 ***+1
00052 0363      ; FLAG ***+1
00053 0364      ; EDGES ***+1
00054 0365      ; ERROR ***+2
00055 0367      ; MODE ***+1
00056 0368      ; COUNT ***+2
00057 036A      ;
00058 036A      ;
00059 036A      ; **ORIGIN
00060 026F      ;
00061 026F      ; *****
00062 026F      ;
00063 026F      ; SCALE ALL Y COORDINATES FROM 0..239
00064 026F      ; PSEUDO-COORDINATE RANGE TO 0..199
00065 026F      ; TRUE SCREEN COORDINATE RANGE
00066 026F      ;
00067 026F A0 06      ; SCALE LDY #6
00068 0271 A9 05      ; LDA #D5
00069 0273 85 AC      ; STA MLPCND
00070 0275 89 4A 03    ; STA YMIN,Y
00071 0278 85 AD      ; STA MLPLER
00072 027A 20 11 C0    ; JSR MULT
00073 027D 99 4C 03    ; STA YMIN,Y
00074 0280 88          ; DEY
00075 0281 88          ; DEY
00076 0282 88          ; DEY
00077 0283 10 F0      ; BPL SCLP
00078 0285 60          ; RTS
00079 0286          ;
00080 0286          ;
00081 0286          ; *****
00082 0286          ;
00083 0286          ; EXCHANGE 'MIN' AND 'MID' COORDINATES
00084 0286          ;
00085 0286 A0 02      ; SWAP12 LDY #2
00086 0288 89 4A 03    ; LOOP1 LDA XMIN,Y
00087 028B 48          ; PHA
00088 028C 89 4D 03    ; LDA XMID,Y
00089 028F 99 4A 03    ; STA XMIN,Y
00090 0292 68          ; PLA
00091 0293 99 4D 03    ; STA XMID,Y
00092 0296 88          ; DEY
00093 0297 10 EF      ; BPL LOOP1
00094 0299 60          ; RTS
00095 029A          ;
00096 029A          ; *****
00097 029A          ;
00098 029A          ; EXCHANGE 'MID' AND 'MAX' COORDINATES
00099 029A          ;
00100 029A A0 02      ; SWAP23 LDY #2
00101 029C 89 4D 03    ; LOOP2 LDA XMID,Y
00102 029F 48          ; PHA
00103 02A0 89 50 03    ; LDA XMAX,Y
00104 02A3 99 4D 03    ; STA XMID,Y
00105 02A6 68          ; PLA
00106 02A7 99 50 03    ; STA XMAX,Y
00107 02AA 88          ; DEY
00108 02AB 10 EF      ; BPL LOOP2
00109 02AD 60          ; RTS
00110 02AE          ;
00111 02AE          ; *****
00112 02AE          ;
00113 02AE          ; SORT COORDINATES ACCORDING TO X COMPONENTS
00114 02AE          ;
00115 02AE A2 02      ; SORTX LDY #2
00116 02B0 38          ; SORTLP SEC
00117 02B1 AD 4D 03    ; LDA XMID
00118 02B4 ED 4A 03    ; SBC XMIN
00119 02B7 AD 4E 03    ; LDA XMID+1
00120 02BA ED 4B 03    ; SBC XMIN+1
00121 02BD 80 03      ; BCS NOSWP1
00122 02BF 20 86 C2    ; JSR SWAP12
00123 02C2 CA          ; NOSWP1 DEX
00124 02C3 F0 15      ; BEQ SORTED
00125 02C5 38          ; SEC
00126 02C6 AD 50 03    ; LDA XMAX
00127 02C9 ED 4D 03    ; SBC XMID
00128 02CC AD 51 03    ; LDA XMAX+1
00129 02CF ED 4E 03    ; SBC XMID+1
00130 02D2 80 DC      ; BCS SORTLP
00131 02D4 20 9A C2    ; JSR SWAP23
00132 02D7 4C 50 C2    ; JMP SORTLP
00133 02DA 60          ;
00134 02DB          ; SORTED RTS
00135 02DB          ; *****
00136 02DB          ;
00137 02DB          ; DRAW A LINE BETWEEN XMIN,YMIN AND XMID,YMID
00138 02DB          ; USING FAST DDA (DIGITAL DIFFERENTIAL ANALYZER)
00139 02DB          ; TECHNIQUE
00140 02DB          ;
00141 02DB A9 02      ; LINE LDY #2
00142 02DD 8D 51 03    ; STA XMAX+1
00143 02DE 20 AE C2    ; JSR SORTX
00144 02E3 AD 47 03    ; LDA NGSCAL
00145 02E6 F0 03      ; BEQ OUTLN
00146 02E8 20 6F C2    ; JSR SCALE
00147 02EB 20 6F C4    ; OUTLN JSR FINDXY
00148 02EE AD 4A 03    ; LDA XMIN
00149 02F1 8D 3F 03    ; STA XPLT
00150 02F4 AD 4B 03    ; LDA XMIN+1
00151 02F7 8D 40 03    ; STA XPLT+1
00152 02FA AD 4C 03    ; LDA YMIN
00153 02FD 8D 41 03    ; STA YPLT
00154 0300 AD 57 03    ; LDA DLTA1+1
00155 0303 D0 7D      ; BNE STEPX
00156 0305 38          ; SEC
00157 0306 AD 56 03    ; LDA DLTA1
00158 0309 ED 5B 03    ; SBC DLTA1
00159 030C 80 74      ; BCS STEPX
00160 030E AD 5B 03    ; LDA DLTA1
00161 0311 8D 65 03    ; STA ERROR
00162 0314 8D 6B 03    ; STA COUNT
00163 0317 4E 65 03    ; LSR ERROR
00164 031A 38          ; SEC
00165 031B AD 56 03    ; LDA DLTA1
00166 031E ED 65 03    ; SBC ERROR
00167 0321 8D 65 03    ; STA ERROR
00168 0324 AD 57 03    ; LDA DLTA1+1
00169 0327 E9 00      ; SBC #0
00170 0329 8D 66 03    ; STA ERROR+1
00171 032C EE 68 03    ; INC COUNT
00172 032F AD 67 03    ; LDA MODE
00173 0332 D0 86      ; BNE ERASE1
00174 0334 20 43 C1    ; JSR PLOT
00175 0337 4C 3D C3    ; JMP SK1
00176 033A 20 46 C1    ; ERASE1 JSR UNPLOT
00177 033D AD 60 03    ; LDA FLAG1
00178 0340 D0 05      ; BNE NSLOPE
00179 0342 EE 41 03    ; INC YPLT
00180 0345 D0 03      ; BNE SK2
00181 0347 CE 41 03    ; NSLOPE DEC YPLT
00182 034A 2C 66 03    ; SK2 BIT ERROR+1
00183 034D 30 1A      ; BMI SK3
00184 034F EE 3F 03    ; INC XPLT
00185 0352 D0 03      ; BNE NOINC1
00186 0354 EE 40 03    ; INC XPLT+1
00187 0357 38          ; NOINC1 SEC
00188 0358 AD 65 03    ; LDA ERROR
00189 035B ED 5B 03    ; SBC DLTA1
00190 035E 8D 65 03    ; STA ERROR
00191 0361 AD 66 03    ; LDA ERROR+1
00192 0364 E9 00      ; SBC #0
00193 0366 8D 66 03    ; STA ERROR+1
00194 0369 18          ; SK3 CLC
00195 036A AD 65 03    ; LDA ERROR
00196 036D 6D 56 03    ; ADC DLTA1
00197 0370 8D 65 03    ; STA ERROR
00198 0373 AD 66 03    ; LDA ERROR+1
00199 0376 6D 57 03    ; ADC DLTA1+1
00200 0379 8D 66 03    ; STA ERROR+1
00201 037C CE 6B 03    ; DEC COUNT
00202 037F D0 AE      ; BNE LNLPI
00203 0381 60          ; RTS
00204 0382          ;
00205 0382 AD 56 03    ; STEPX LDA DLTA1
00206 0385 8D 65 03    ; STA ERROR
00207 0388 8D 6B 03    ; STA COUNT
00208 038B AD 57 03    ; LDA DLTA1+1
00209 038E 8D 66 03    ; STA ERROR+1
00210 0391 8D 69 03    ; STA COUNT+1
00211 0394 4E 66 03    ; LSR ERROR+1
00212 0397 6E 65 03    ; ROR ERROR
00213 039A 38          ; SEC
00214 039B AD 5B 03    ; LDA DLTA1
00215 039E ED 65 03    ; SBC ERROR
00216 03A1 8D 65 03    ; STA ERROR
00217 03A4 A9 00      ; LDA #0
00218 03A6 ED 66 03    ; SBC ERROR+1
00219 03A9 8D 66 03    ; STA ERROR+1
00220 03AC AD 67 03    ; LDA MODE
00221 03AF D0 06      ; BNE ERASE2
00222 03B1 20 43 C1    ; JSR PLOT
00223 03B4 4C BA C3    ; JMP SKP1
00224 03B7 20 46 C1    ; ERASE2 JSR UNPLOT
00225 03BA EE 3F 03    ; SKP1 INC XPLT
00226 03BD D0 03      ; BNE NOINC2
00227 03BF EE 40 03    ; INC XPLT+1
00228 03C2 2C 66 03    ; NOINC2 BIT ERROR+1
00229 03C5 30 20      ; BMI SKP3
00230 03C7 AD 60 03    ; LDA FLAG1
00231 03CA D0 05      ; BNE NGSLP
00232 03CC EE 41 03    ; INC YPLT
00233 03CF D0 03      ; BNE SKP2
00234 03D1 CE 41 03    ; NGSLP DEC YPLT
00235 03D4 38          ; SKP2 SEC
00236 03D5 AD 65 03    ; LDA ERROR
00237 03D8 ED 5B 03    ; SBC DLTA1
00238 03DB 8D 65 03    ; STA ERROR
00239 03DE AD 66 03    ; STA ERROR+1
00240 03E1 ED 57 03    ; LDA ERROR+1
00241 03E4 8D 66 03    ; STA ERROR+1
00242 03E7 18          ; SKP3 CLC
00243 03E8 AD 65 03    ; LDA ERROR
00244 03EB 6D 5B 03    ; ADC DLTA1
00245 03EE 8D 65 03    ; STA ERROR
00246 03F1 AD 66 03    ; LDA ERROR+1
00247 03F4 69 00      ; ADC #0
00248 03F6 8D 66 03    ; STA ERROR+1
00249 03F9 38          ; SEC
00250 03FA AD 6B 03    ; LDA COUNT
00251 03FD E9 01      ; SBC #1
00252 03FF 8D 6B 03    ; STA COUNT
00253 0402 80 03      ; BCS TEST
00254 0404 CE 69 03    ; DEC COUNT+1
00255 0407 2C 69 03    ; BIT COUNT+1
00256 040A 10 A0      ; BPL LNLPI
00257 040C C40C      ; RTS
00258 040D          ;
00259 040D          ; *****
00260 040D          ; DRAW A SHADED VERTICAL LINE AT
00261 040D          ; XPLT FROM YTOP TO YBOT
00262 040D          ;
00263 040D 38          ; VLINE SEC
00264 040E AD 53 03    ; LDA YTOP
00265 0411 ED 54 03    ; SBC YBOT
00266 0414 80 0E      ; BCS DRAW
00267 0416 AD 53 03    ; LDA YTOP
00268 0419 48          ; PHA
00269 041A AD 54 03    ; LDA YBOT
00270 041D 8D 53 03    ; STA YBOT
00271 0420 68          ; PLA
00272 0421 8D 54 03    ; STA YBOT

```

(Continued on page 68)

De Smet C

8086/8088 Development Package

\$109

FULL DEVELOPMENT PACKAGE

- Full K&R C Compiler
- Assembler, Linker & Librarian
- Full Screen Editor
- Execution Profiler
- Complete STDIO Library (>120 Func)

Automatic DOS 1.X/2.X SUPPORT

BOTH 8087 & S/W FLOATING POINT

OVERLAYS

OUTSTANDING PERFORMANCE

- First and Second in AUG '83 BYTE benchmarks

SYMBOLIC DEBUGGER \$50

- Examine & change variables by name using C expressions
- Flip between debug and display screen
- Display C source during execution
- Set multiple breakpoints by function or line number

DOS LINK SUPPORT \$35

- Converts DeSmet.O to DOS.OBJ Format
- LINKS with DOS ASM
- Uses Lattice® naming conventions

CWARE

CORPORATION

P.O. Box C, Sunnyvale, CA 94087
(408) 720-9696

Street Address: 505 W. Olive, #767 (94086) Call for hrs.

All orders shipped UPS surface on IBM format disks. Shipping included in price. California residents add sales tax. Canada shipping add \$5, elsewhere add \$15. Checks must be on U.S. Bank and in U.S. Dollars. Call 9am-1pm to CHARGE by VISA/MC/AMEX.

Foreign Distributors: **AFRICA**, HI-TECH SVCS, Gaborone 4540 or Telex 2205BD LANGER • **ENGLAND**: MLH Tech, 0606-891146 • **JAPAN**: JSE 03-486-7151 • **SWEDEN**: ESCORT DATA 08-87 41 48 or THESEUS KONSULT 08-23 61 60

Circle no. 18 on reader service card.

FINALLY...

THE C JOURNAL

The C Journal will help YOU use C on YOUR machine — IBM PC™, CP/M™, Macintosh™, or UNIX™-based — micro, mini, or mainframe.

It's the ONE publication for programmers, software managers, and other computer professionals who need to keep aware of developments in the industry's fastest-growing language.

- regular columns for novice through advanced C programmers
- software product and book reviews
- tips on working with major compilers and operating systems
- news from the ANSI standards committee and the industry

For **FREE** sample issue and discount subscription information, write, call, or circle our reader service number. The C Journal is a quarterly publication, and costs \$28/year (add \$9 for overseas airmail).

Subscriptions/Advertising:
Christina Gardner
(201) 989-0570

Editorial:
Rex Jaeschke
(703) 860-0091

another independent publication from



InfoPro Systems

3108 Route 10
P.O. Box 849
Denville, NJ 07834



Circle no. 91 on reader service card.

6 TIMES FASTER!

SuperFast Software Development Tools

INCREASE YOUR PROGRAMMING EFFICIENCY

with high-performance software development products from SLR Systems.

No other tools approach the speed or flexibility of the SLR Systems line.

"Z80ASM is an extraordinary product..."
Robert Blum, Sept. 84 DDJ

"...in two words, I'd say speed & flexibility",
Edward Joyce, Nov. 84 Microcomputing

ASSEMBLERS

- RMAC/M80 macros
- Nested INCLUDES & conditionals
- 16 char. labels on externals
- Built in cross-reference
- Optional case significance
- Phase/dephase
- Math on external words and bytes
- Define symbols from console
- Generate COM, HEX, SLR-REL, or Micro-soft-REL files
- Time & Date in listing
- Over 30 configure options

Z80ASM -full Zilog Z80 \$125

NEW! Z80ASM+ -all tables virtual \$195

NEW! SLRMAC -full Intel 8080, with

Z80.LIB extensions internal \$125

NEW! SLRMAC+ -all tables virtual \$195

Z80 CPU, CP/M compatible, 32K TPA required.

"Z80ASM...a breath of fresh air..."
Computer Language, Feb. 85



C.O.D., Check or Money Order Accepted

LINKERS

- Links SLR & M80 format files
- Output HEX or COM file
- Three separate address spaces
- Load map and SID/ZSID .SYM file
- SLRINK+ includes:
 - All tables overflow to disk
 - HEX files do not fill unused space
 - Intermodule cross-reference
 - EIGHT separate address spaces
 - Works with FORTRAN & BASIC
 - Generate PRL & SPR files
 - Supports manual overlays
 - Full 64K output

SLRINK -fastest memory based \$125

NEW! SLRINK+ -full featured virtual \$195

Combo Paks available from \$199. - \$299.

For additional information contact SLR Systems

1-800-833-3061, in PA (412) 282-0864
1622 N. Main St., Butler, PA 16001 • Telex 559215

SLR Systems

Circle no. 78 on reader service card.

Listing Three

```

00273 C424 AD 53 03 DRAW LDA YTOP
00274 C427 BD 41 03 STA YPLT
00275 C42A 20 24 C2 JSR NORM ; PLOT A SHADE-WEIGHTED
00276 C42D AD 53 03 LDA YTOP ; PIXEL CHECKING ONLY
00277 C430 CD 54 03 CMP YBOT ; FOR SHADE STYLE
00278 C433 F8 06 BEQ DONE
00279 C435 2E 53 03 DEC YTOP
00280 C438 4C 24 C4 JMP DRAW
00281 C43B 60 RTS
00282 C43C ;
00283 C43C ;
00284 C43C ;
00285 C43C ;
00286 C43C ;
00287 C43C ;
00288 C43C AD 5F 03 DONE RTS
00289 C43F 85 AC ;
00290 C441 AD 5E 03 ;
00291 C444 85 AD ;
00292 C446 20 11 C0 ;
00293 C449 85 FE ;
00294 C44B A5 AE ;
00295 C44D 85 FD ;
00296 C44F A9 00 ;
00297 C451 85 FC ;
00298 C453 AD 5A 03 ;
00299 C456 85 FB ;
00300 C458 20 25 C0 ;
00301 C45B AD 63 03 ;
00302 C45E D0 09 ;
00303 C460 18 ;
00304 C461 AD 55 03 ;
00305 C464 65 FD ;
00306 C466 98 06 ;
00307 C468 38 ;
00308 C469 AD 55 03 ;
00309 C46C E5 FD ;
00310 C46E 60 ;
00311 C46F ;
00312 C46F ;
00313 C46F ;
00314 C46F ;
00315 C46F ;
00316 C46F ;
00317 C46F ;
00318 C46F ;
00319 C46F 38 ;
00320 C470 AD 40 03 ;
00321 C473 ED 4A 03 ;
00322 C476 BD 56 03 ;
00323 C479 AD 4E 03 ;
00324 C47C ED 4B 03 ;
00325 C47F BD 57 03 ;
00326 C482 38 ;
00327 C483 AD 50 03 ;
00328 C486 ED 4D 03 ;
00329 C489 BD 58 03 ;
00330 C48C 38 ;
00331 C48D AD 50 03 ;
00332 C490 ED 4A 03 ;
00333 C493 BD 59 03 ;
00334 C496 ;
00335 C496 ;
00336 C496 ;
00337 C496 ;
00338 C496 A9 00 ;
00339 C49B BD 60 03 ;
00340 C49B BD 61 03 ;
00341 C49E BD 62 03 ;
00342 C4A1 38 ;
00343 C4A2 AD 4F 03 ;
00344 C4A5 ED 4C 03 ;
00345 C4A8 D0 09 ;
00346 C4AA EE 60 03 ;
00347 C4AD AD 4C 03 ;
00348 C4B0 ED 4F 03 ;
00349 C4B3 BD 5B 03 ;
00350 C4B6 38 ;
00351 C4B7 AD 52 03 ;
00352 C4BA ED 4F 03 ;
00353 C4BD D0 09 ;
00354 C4BF EE 61 03 ;
00355 C4C2 AD 4F 03 ;
00356 C4C5 ED 52 03 ;
00357 C4C8 BD 5C 03 ;
00358 C4CB 38 ;
00359 C4CC AD 52 03 ;
00360 C4CF ED 4C 03 ;
00361 C4D2 D0 09 ;
00362 C4D4 EE 62 03 ;
00363 C4D7 AD 4C 03 ;
00364 C4DA ED 52 03 ;
00365 C4DD BD 5D 03 ;
00366 C4E0 60 ;
00367 C4E1 ;
00368 C4E1 ;
00369 C4E1 ;
00370 C4E1 ;
00371 C4E1 ;
00372 C4E1 20 AE C2 ;
00373 C4E4 AD 47 03 ;
00374 C4E7 F0 03 ;
00375 C4E9 20 6F C2 ;
00376 C4EC 20 6F C4 ;
00377 C4EF AD 4A 03 ;
00378 C4F2 BD 3F 03 ;
00379 C4F5 AD 4B 03 ;
00380 C4F8 BD 40 03 ;
00381 C4FB 38 ;
00382 C4FC AD 3F 03 ;
00383 C4FF ED 4A 03 ;
00384 C502 BD 5F 03 ;
00385 C505 AD 56 03 ;
00386 C508 F0 53 ;
00387 C50A BD 5A 03 ;
00388 C50D AD 56 03 ;
00389 C510 BD 5E 03 ;
00390 C513 AD 60 03 ;
00391 C516 BD 63 03 ;
00392 C519 AD 4C 03 ;
00393 C51C BD 55 03 ;
00394 C51F 20 3C C4 ;
00395 C522 BD 53 03 ;
00396 C525 AD 59 03 ;
00397 C528 F0 33 ;
00398 C52A BD 5A 03 ;
00399 C52D AD 5D 03 ;
00400 C530 BD 5E 03 ;

```

```

00401 C533 AD 62 03 LDA FLAG3
00402 C536 BD 63 03 STA FLAG
00403 C539 20 3C C4 JSR ENDP15
00404 C53C BD 54 03 STA YBOT
00405 C53F 20 0D C4 JSR VLINE
00406 C542 AD 40 03 LDA XPLT+1
00407 C545 CD 4E 03 CMP XMID+1
00408 C548 D0 08 BNE NEXTX1
00409 C54A AD 3F 03 LDA XPLT
00410 C54D CD 4D 03 CMP XMID
00411 C550 F0 08 BEQ CONT
00412 C552 EE 3F 03 NEXTX1 INC XPLT
00413 C555 D0 03 BNE SKIP3
00414 C557 EE 40 03 INC XPLT+1
00415 C55A 4C FB C4 SKIP3 JMP FCETLP
00416 C55D 38 CONT
00417 C55E AD 3F 03 LDA XPLT
00418 C561 ED 4A 03 SBC XMID
00419 C564 BD 5F 03 STA XDIF
00420 C567 AD 59 03 LDA DLTA3
00421 C56A F0 63 BEQ FINI
00422 C56C BD 5A 03 STA DELTA3
00423 C56F AD 5D 03 LDA DLTA3
00424 C572 BD 5E 03 STA DELTA3
00425 C575 AD 62 03 STA FLAG3
00426 C578 BD 63 03 STA FLAG
00427 C57B AD 4C 03 LDA YMIN
00428 C57E BD 55 03 STA YBASE
00429 C581 20 3C C4 JSR ENDP15
00430 C584 BD 54 03 STA YBOT
00431 C587 38 SEC
00432 C588 AD 3F 03 LDA XPLT
00433 C58B ED 4D 03 SBC XMID
00434 C58E BD 5F 03 STA XDIF
00435 C591 AD 5B 03 LDA DLTA2
00436 C594 F0 39 BEQ FINI
00437 C596 BD 5A 03 STA DELTA2
00438 C599 AD 5C 03 LDA DLTA2
00439 C59C BD 5E 03 STA DELTA2
00440 C59F AD 61 03 LDA FLAG2
00441 C5A2 BD 63 03 STA FLAG
00442 C5A5 AD 4F 03 LDA YHID
00443 C5A8 BD 55 03 STA YBASE
00444 C5AB 20 3C C4 JSR ENDP15
00445 C5AE BD 53 03 STA YTOP
00446 C5B1 20 0D C4 JSR VLINE
00447 C5B4 AD 40 03 LDA XPLT+1
00448 C5B7 CD 51 03 CMP XMID+1
00449 C5BA D0 08 BNE NEXTX2
00450 C5BD AD 3F 03 LDA XPLT
00451 C5BF CD 50 03 CMP XMID
00452 C5C2 F0 08 BEQ FINI
00453 C5C4 EE 3F 03 NEXTX2 INC XPLT
00454 C5C7 D0 03 BNE SKIP4
00455 C5C9 EE 40 03 INC XPLT+1
00456 C5CC 4C 5D C5 SKIP4 JMP CONT
00457 C5CF AD 64 03 FINI LDA EDGES
00458 C5D2 F0 15 BEQ FINISH
00459 C5D4 20 EB C2 JSR OUTLN
00460 C5D7 20 9A C2 JSR SWAP23
00461 C5DA 20 EB C2 JSR OUTLN
00462 C5DD 20 9A C2 JSR SWAP12
00463 C5E0 20 9A C2 JSR SWAP23
00464 C5E3 20 9A C2 JSR SWAP12
00465 C5E6 20 EB C2 JSR OUTLN
00466 C5E9 60 FINISH RTS
00467 C5EA .END

```

ERRORS = 00000

SYMBOL TABLE

SYMBOL VALUE		COUNT		COUNT		COUNT		COUNT	
CONT	C55D	COUNT	0368	DELTA3	035A	DELTA3	035E		
DIVIDE	C025	DLTA1	0356	DLTA2	0358	DLTA3	0359		
DLTA1	035B	DLTA2	035C	DLTA3	035D	DONE	C43B		
DRAW	C424	DVDND	00FD	DVSOR	00FB	EDGES	036A		
ENDPTS	C43C	ERASE1	C33A	ERASE2	C3B7	ERROR	0365		
FACET	C4E1	FCETLP	C4FB	FINDXY	C46F	FINI	C5CF		
FINISH	C5E9	FLAG	0363	FLAG1	0360	FLAG2	0361		
FLAG3	0362	LINE	C20B	LNL1	C32F	LNL2	C3AC		
LOOP1	C2B8	LOOP2	C29C	MLPCND	00AC	MLPLR	00AD		
MODE	0367	MULT	C011	NEGSLP	C469	NEXTX1	C552		
NEXTX2	C5C4	NGSLP	C3D1	NOINC1	C357	NOINC2	C3C2		
NORM	C22A	NOSCAL	0347	NOISW1	C2C2	NSLOPE	C347		
ORIGIN	C2AF	OUTLN	C2EB	PLOT	C143	PROG	00AE		
QUOT	00FD	RAM	034A	SCALE	C26F	SCLP	C275		
SK1	C3D0	SK2	C34A	SK3	C369	SKIP2	C46E		
SKIP3	C55A	SKIP4	C5CC	SKP1	C3BA	SKP2	C3DA		
SKP3	C3E7	SORTED	C2DA	SORTLP	C2B0	SORTX	C2AE		
STEPX	C3B2	STEPY	C30E	STORE1	C4B3	STORE2	C4C8		
STORE3	C4DD	SWAP12	C2B6	SWAP23	C29A	TEST	C407		

SYMBOL TABLE

SYMBOL VALUE		COUNT		COUNT		COUNT		COUNT	
UNPLOT	C146	VLINE	C40D	XDIF	035F	XMID	0350		
YHID	034D	XMIN	034A	XPLT	034F	YBASE	0355		
YBOT	0354	YMAX	0352	YHID	034F	YMIN	034C		
YPLT	0341	YSOK	C4EC	YTOP	0353				

END OF ASSEMBLY

End Listing Three

Listing Four

```

00001 0000 ; PRIMITIVE SOLID SHAPE DRAWING
00002 0000 ;
00003 0000 ; RICHARD L. RYLANDER 11/7/84
00004 0000 ;
00005 0000 ; LOAD ARITHMETIC AND GRAPHIC UTILITIES FIRST
00006 0000 ;
00007 0000 ;
00008 0000 ;
00009 0000 ;
00010 0000 ;
00011 0000 ;
00012 0000 ;
00013 0000 ;
00014 0000 ;
00015 0000 ;
00016 0000 ;

```

(Continued on page 70)

ENVOY™

Communications Software

- Easy to use, menu driven, compact and high speed
- Access electronic mail, remote systems and data networks like CompuServe, The Source and Dow Jones News Retrieval
- Terminal mode with large data-capture buffer
- Error-free transfers of text and binary files
- XMODEM (Christensen) and ANSI X3.28 transfer protocols
- User-definable Autodial and Autologin menu
- Utilities menu for file copy, type, print, erase and rename
- Remote unattended file transfers and utilities
- Available for most popular computer systems: IBM PC, PCjr, PC compatibles, Sanyo MBC-55X, MSDOS, CP/M-86, Concurrent CP/M, CP/M 2.2, CP/M Plus and more
- Money-back guarantee if not completely satisfied

\$49⁹⁵

ARTISOFT INC

P.O. Box 41436
Tucson, AZ 85717
(602) 327-4305

ConIX™

NOW ONLY \$79.95!

If you think you're missing out on innovative software developments because nobody is writing for CP/M™.80, take a look at us. We've adapted UNIX™ features to CP/M like never before, and with the kind of professional, quality-controlled product that you deserve. That product is none other than the critically acclaimed ConIX Operating System.

ConIX can provide any 48K+ CP/M-80 or compatible system with I/O Redirection and Pipes (uses memory or disk), perfected User Areas, Command and Overlay Path Searching, Auto Screen Paging, 8Mb Print Buffering, 22 new SysCalls, Function Keys, "Virtual" disk system, Archiver (saves over 50% disk), extensive command language, 300+ variables, 100+ commands, pull-down menu, and much more! Uses as little as 1/2K RAM! Runs with CP/M for true data and software compatibility. Installs easily without any system mods!

The ConIX package lists at \$165 and has been advertised and sold internationally to many enthusiastic customers since October 1983. As a special limited offer, we've lowered the price of the complete ConIX system by 50% to only \$79.95! Don't miss this opportunity to bring your 8-bit micro back into the software revolution. Order your copy of ConIX today!

Price includes manual, 8" disk, and user support. 5¼" conversions available. Contact your local dealer, or buy direct and add shipping: \$4.50 UPS, \$10 Canada, \$25 overseas. NY residents add sales tax.



Computer Helper Industries Inc.
P.O. Box 680 Parkchester Station, NY 10462
Tel. (212) 652-1786 (for information/orders)

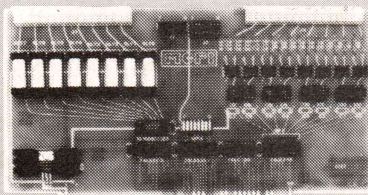
"We're helping your computer work better for you!"

UNIX: AT&T Bell Labs, CP/M: Digital Research, ConIX: Computer Helper Ind.

Circle no. 7 on reader service card.

Circle no. 22 on reader service card.

MULLEN S-100: Real Time, Real World **CONTROLLER**



ICB-10 CONTROLLER BOARD
\$219, assembled and tested.

This 8 channel digital I/O controller can monitor inputs and control outputs. It features an easy to read manual that has schematics, component list, and programming examples as well as provocative insights on potential applications.

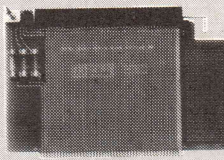
Examples of applications are included in a reprinted article that demonstrates two MULLEN CONTROLLER BOARDS in an interactive system that: feeds a cat; irrigates a garden dependent on soil moisture; closes the window when it rains; controls the thermostat for optimum comfort; controls appliances, lights, security system, and weather monitoring station (logging temperature, wind speed and direction, and graphing pollution content of the atmosphere.) Solenoids, microswitches, pneumatic actuators, pH sensors, and other devices are used in this system.

MULLEN S-100: **DEBUGGERS**



TB-4a EXTENDER BOARD
The latest in our TB line, the most widely used add-ons in the industry. Features logic probe, formed-lead edge connectors, pulse catcher switch and reset button.

\$89, assembled and tested.



ZB-1 ZIF EXTENDER BOARD
This debugger features Zero Insertion Force edge connectors for easy board changes and long life. Expect 2,000 or more insertions rather than the usual 300 to 400 with tension type connectors.

\$159, assembled and tested.



MULLEN COMPUTER PRODUCTS, INC.

AVAILABLE:

Priority One Electronics,
Chatworth, CA • (213) 709-5111
Jade Computer Products
Hawthorne, CA • (213) 973-7707
EPI Computer Products
Hayward, CA • (415) 786-9203

Mullen Computer Products is the industrial distributor for CompuPro's products. For more information, call us at (415) 783-2866 or write MCPI, 2260 American Ave., #1, Hayward, CA 94545. OEM sales available from factory. Prices are subject to change without notice.

Circle no. 54 on reader service card.

Drawing on the C-64 (Listing Continued, text begins on page 50)

Listing Four

```

00017 0000      DVSOR=#FB      ; DIVISOR (D)
00018 0000      QUOT=#FD      ; QUOTIENT (D)
00019 0000      DIVIDE=#C025    ; CALL FOR DIVIDE
00020 0000
00021 0000      ARG=#AC      ; ARGUMENT (S)
00022 0000      SQR=#AE      ; SQUARE OF ARG (D)
00023 0000      SQUARE=#C004    ; CALL FOR SQUARE
00024 0000
00025 0000      RADCND=#AC      ; RADICAND (D)
00026 0000      ROOT=#033C      ; SQUARE ROOT (S)
00027 0000      SQRRT=#C064    ; CALL FOR SQRRT
00028 0000
00029 0000      RNDM=#C000      ; RANDOM NUMBER
00030 0000      RANDOM=#C0CB      ; CALL FOR RANDOM
00031 0000
00032 0000      ; NOTE - A CALL TO 'RANDOM' LEAVES A RANDOM BYTE
00033 0000      ; IN THE ACCUMULATOR
00034 0000
00035 0000      XPLT=#033F      ;
00036 0000      YPLT=#0341      ;
00037 0000      NORM=#C224      ;
00038 0000      PLTSHD=#C20F      ;
00039 0000      VALUE=#0344      ; FINAL NORMALIZED SHADE VALUE
00040 0000      HTORRN=#0346      ; SHADE FLAG, 1=HALFTONE
00041 0000      NSCAL=#0347      ; SCALE FLAG, 1=NO SCALE
00042 0000
00043 036A      XCENT ***+2      ; CENTER COORD
00044 036C      XREL ***+1      ; RELATIVE (TO CENTER)
00045 036D      XSHD ***+2      ; USED IN SHADE CALC
00046 036F      YCENT ***+1      ; CENTER COORD
00047 0370      YREL ***+1      ; RELATIVE (TO CENTER)
00048 0371      YSHD ***+2      ; USED IN SHADE CALC
00049 0373      ZREL ***+2      ; RELATIVE (TO CENTER)
00050 0375      ZWX ***+2      ; Z WITH X (+ OR -)
00051 0377
00052 0377      RADIUS ***+2      ; LOCAL RADIUS OF SURFACE
00053 0379      TONE ***+2      ; USED IN SHADE CALC
00054 037B      TINTMP ***+2      ; USED IN SHADE CALC
00055 037D
00056 037D      CLIPL ***+1      ; LEFT CLIPPING BOUND
00057 037E      CLIPR ***+1      ; RIGHT CLIPPING BOUND
00058 037F      CLIPU ***+1      ; UP CLIPPING BOUND
00059 0380      CLIPD ***+1      ; DOWN CLIPPING BOUND
00060 0381
00061 0381      HEMI ***+1      ; PLOTTING HEMISPHERE
00062 0382
00063 0382      BAKLIT ***+1      ; BACKLIT FLAG
00064 0383      HVFLAG ***+1      ; HORIZONTAL/VERTICAL FLAG
00065 0384      TEMP ***+2      ; TEMPORARY STORAGE
00066 0386      CNTX ***+1      ; LOOP COUNTER
00067 0387      CNTY ***+1      ; LOOP COUNTER
00068 0388      MAX ***+1      ; LOOP LIMIT
00069 0389
00070 0389      HLEN ***+1      ; HALF LENGTH OF CYLINDERS
00071 038A      RS ***+2      ; SQUARE OF TOROID RADIUS
00072 038C      RI ***+1      ; TOROID (RING) RADIUS
00073 038D      RC ***+1      ; CORNER RADIUS OF TOROID
00074 038E      RO ***+1      ; OUTER RADIUS OF TOROID
00075 038F      RI ***+1      ; INNER RADIUS OF TOROID
00076 0390      XSOR ***+2      ;
00077 0392      XMAX ***+1      ;
00078 0393
00079 0393      R0=HLEN
00080 0393
00081 0393      *-ORIGIN
00082 C5EA
00083 C5EA      ; *****
00084 C5EA      ; DIVIDE WITH SINGLE PRECISION DIVISOR
00085 C5EA      ; (USED OFTEN IN SHAPE ROUTINES)
00086 C5EA
00087 C5EA A9 00      LDIV LDA #0
00088 C5EC 85 FC      STA DVSOR+1
00089 C5EE 4C 25 C0    JMP DIVIDE
00090 C5F1
00091 C5F1      ; *****
00092 C5F1      ; CALCULATE SHADE VALUE (0-63) BY
00093 C5F1      ; MULTIPLYING 'TONE' BY 26 THEN
00094 C5F1      ; DIVIDE RESULT BY RADIUS OF SURFACE
00095 C5F1
00096 C5F1      GETVAL BIT TONE+1
00097 C5F1      BPL CNTNU      ; IF 'TONE' < 0, THEN
00098 C5F4 10 12      LDA BAKLIT      ; MAKE VALUE 0 OR ABS(TONE)
00099 C5F6 AD 82 03    LDA BAKLIT      ; DEPENDING ON BAKLIT FLAG
00100 C5F9 D0 04      BNE NEGATE
00101 C5FB 80 44 03    STA VALUE
00102 C5FE 60      RTS
00103 C5FF 30      NEGATE SEC
00104 C600 A9 00      LDA #000
00105 C602 ED 79 03    SBC TONE
00106 C605 8D 79 03    STA TONE
00107 C608 AD 79 03    LDA TONE
00108 C60B 85 AC      CNTNU LDA TONE
00109 C60D A9 1A      LDA #1A
00110 C60F 85 AD      STA #1A
00111 C611 20 11 C0    STA MLPLR
00112 C614 85 FE      JSR MULT
00113 C616 A5 AE      STA DVOND+1
00114 C618 85 FD      LDA FROD
00115 C61A AD 77 03    STA DVOND
00116 C61D 85 FB      LDA RADIUS
00117 C61F 20 6A C5    STA DVSOR
00118 C622 A5 FD      JSR SDIV
00119 C624 8D 44 03    LDA QUIT
00120 C627 60      STA VALUE
00121 C628      RTS
00122 C628
00123 C628      ; *****
00124 C628      ; POINT PLOTTING BY QUADRANTS USING
00125 C628      ; THE FOUR-FOLD SYMMETRY OF SIMPLE OBJECTS
00126 C628
00127 C628      ; DEPENDING ON STATUS OF 'HVFLAG', EXCHANGE
00128 C628      ; X AND Y COORDINATES TO ROTATE OBJECTS 90 DEG
00129 C628      ; SINGLE SHAPE ROUTINE CAN THEN BE USED TO
00130 C628      ; DRAW 'HORIZONTAL' OR 'VERTICAL' VERSIONS
00131 C628      ; OF AN OBJECT
00132 C628
00133 C628      ; THE FOLLOWING IS A 'BASIC SUBROUTINE'
00134 C628      ; EQUIVALENT TO EXPLAIN ITS OPERATION
00135 C628
00136 C628      ; NOTE THAT LABELS ARE USED IN PLACE OF
00137 C628      ; LINE NUMBERS
00138 C628
00139 C628      ; 'PTPLOT' IF HVFLAG<0 THEN GOTO 'NOROT'
00140 C628      ; (STACK)=XSHD:XSHD=YSHD:YSHD=(STACK)
00141 C628      ; (STACK)=XSHD:XSHD=YSHD:YSHD=(STACK)
00142 C628      ; 'NOROT' GOSUB 'GETZ'
00143 C628      ; REM CALCULATE 2*Z FROM X,Y AND RADIUS
00144 C628      ; HEMI = 1
00145 C628      ; IF XREL>CLIPL THEN GOTO 'RHEMI'
00146 C628      ;
00147 C628      ;
00148 C628      ;
00149 C628      ;
00150 C628      ;
00151 C628      ;
00152 C628      ;
00153 C628      ;
00154 C628      ;
00155 C628      ;
00156 C628      ;
00157 C628      ;
00158 C628      ;
00159 C628      ;
00160 C628      ;
00161 C628      ;
00162 C628      ;
00163 C628      ;
00164 C628      ;
00165 C628      ;
00166 C628      ;
00167 C628 2C 83 03    PTPLOT BIT HVFLAG
00168 C628 10 2D      BPL NOROT
00169 C62D AD 6C 03    LDA XREL
00170 C630 48      PHA
00171 C631 48      PHA
00172 C632 AD 70 03    LDA YREL
00173 C635 8D 6C 03    STA XREL
00174 C638 68      PLA
00175 C639 8D 70 03    STA YREL
00176 C63C AD 6D 03    LDA XSHD
00177 C63F 48      PHA
00178 C640 48      PHA
00179 C641 AD 71 03    LDA YSHD
00180 C644 8D 6D 03    STA XSHD
00181 C647 68      PLA
00182 C648 8D 71 03    STA YSHD
00183 C64B AD 6E 03    LDA XSHD+1
00184 C64E 48      PHA
00185 C64F 48      PHA
00186 C650 AD 72 03    LDA YSHD+1
00187 C653 8D 6E 03    STA XSHD+1
00188 C656 68      PLA
00189 C657 8D 72 03    STA YSHD+1
00190 C65A 20 45 C7    NOROT JSR GETZ
00191 C65D A9 01      PTLT2 LDA #01
00192 C65F 8D 81 03    STA HEMI
00193 C662 38      SEC
00194 C663 AD 7D 03    LDA CLIPL      ; CHECK LEFT HEMISPHERE
00195 C666 CD 6C 03    CMP XREL
00196 C669 90 7D      BCC RHEMI
00197 C66B 38      SEC
00198 C66C AD 3C 03    LDA ROOT
00199 C66F ED 6D 03    SBC XSHD
00200 C672 8D 75 03    STA ZWX
00201 C675 AD 3D 03    LDA ROOT+1
00202 C678 ED 6E 03    SBC XSHD+1
00203 C67B 8D 76 03    STA ZWX+1
00204 C67E 38      SEC
00205 C67F AD 6A 03    LDA XCENT
00206 C682 ED 6C 03    SBC XREL
00207 C685 8D 3F 03    STA XPLT
00208 C688 AD 6B 03    LDA XCENT+1
00209 C68B E9 00      SBC #000
00210 C68D 8D 40 03    STA XPLT+1
00211 C690
00212 C690 38      CHCLUP SEC
00213 C691 AD 7F 03    LDA CLIPU      ; CHECK FOR UP CLIPPING
00214 C694 CD 70 03    CMP YREL
00215 C697 90 23      BCC DHEMI
00216 C699 18      CLC
00217 C69A AD 75 03    LDA ZWX
00218 C69D 6D 71 03    ADC YSHD
00219 C6A0 8D 79 03    STA TONE
00220 C6A3 AD 76 03    LDA ZWX+1
00221 C6A6 6D 72 03    ADC YSHD+1
00222 C6A9 8D 7A 03    STA TONE+1
00223 C6AC 20 F1 C5    JSR GETVAL
00224 C6AF 18      CLC
00225 C6B0 AD 6F 03    LDA YCENT
00226 C6B3 6D 70 03    ADC YREL
00227 C6B6 AD 41 03    STA YFLT
00228 C6B9 20 0F C2    JSR FLTSHD
00229 C6BC
00230 C6BC 38      DHEMI SEC
00231 C6BD AD 80 03    LDA CLIPD      ; CHECK FOR DOWN CLIPPING
00232 C6C0 CD 70 03    CMP YREL
00233 C6C3 90 23      BCC RHEMI
00234 C6C5 38      SEC
00235 C6C6 AD 75 03    LDA ZWX
00236 C6C9 ED 71 03    SBC YSHD
00237 C6CC 8D 79 03    STA TONE
00238 C6CF AD 76 03    LDA ZWX+1
00239 C6D2 ED 72 03    SBC YSHD+1
00240 C6D5 8D 7A 03    STA TONE+1
00241 C6D8 20 F1 C5    JSR GETVAL
00242 C6DB 38      SEC
00243 C6DC AD 6F 03    LDA YCENT
00244 C6DF ED 70 03    ADC YREL
00245 C6E2 8D 41 03    STA YFLT
00246 C6E5 20 0F C2    JSR FLTSHD
00247 C6E8
00248 C6E8 AD 81 03      RHEMI LDA HEMI
00249 C6EB F0 34      BEQ PLDONE
00250 C6ED CE 81 03    DEC HEMI
00251 C6F0 38      SEC
00252 C6F1 AD 7E 03    LDA CLIPR      ; CHECK FOR RIGHT CLIPPING
00253 C6F4 CD 6C 03    CMP XREL
00254 C6F7 90 28      BCC PLDONE
00255 C6F9 18      CLC
00256 C6FA AD 3C 03    LDA ROOT
00257 C6FD 6D 6D 03    ADC XSHD
00258 C700 8D 75 03    STA ZWX
00259 C703 AD 3D 03    LDA ROOT+1
00260 C706 6D 6E 03    ADC XSHD+1
00261 C709 8D 76 03    STA ZWX+1
00262 C70C 18      CLC
00263 C70D AD 6A 03    LDA XCENT
00264 C710 AD 6C 03    ADC XREL
00265 C713 8D 3F 03    STA XPLT
00266 C716 AD 6B 03    LDA XCENT+1
00267 C719 69 00      ADC #000
00268 C71B 8D 40 03    STA XPLT+1
00269 C71E 4C 90 C6    JMP CHCLUP
00270 C721 2C 83 03    PLDONE BIT HVFLAG
00271 C724 10 1E      BPL NORSTR

```

(Continued on page 72)

MODEL 100 C COMPILER

Now you can write efficient programs for your TRS-80 model 100 with ease. Or, learn the essentials of C programming while traveling!

C/100 - THE "PORTABLE" C COMPILER

Cassette version \$49.00
Disk/Video interface version \$59.00
Model II version (run on mod II, then download object code to model 100) \$79.00
Model III version (as above for Mod III) \$79.00

Write or call for information on other TRS-80 software.

MODELS II, 12, 16 MODELS III, 4

TRISC C COMPILER

Full K&R with source to the function library. UNIX compatible \$85.00

ZSPF EDITOR

SPF, the choice of most mainframe programmers, is now available for 280 machines. And it's panel driven so you can customize it! \$75.00

business utility software

109 minna ste 423 san francisco ca 94105
(415) 397-2000

C for the Model 200 and SPF for CP/M available soon

Circle no. 8 on reader service card.

PRESENTING THE MEGAMAX C COMPILER

FEATURING:
• IN-LINE ASSEMBLY • ONE PASS
COMPILE • SUPPORT OF DYNAMIC
OVERLAYS • FULL ACCESS OF MACINTOSH
TOOLBOX ROUTINES • AND MUCH MORE ...
DEVELOPMENT SYSTEM PACKAGE INCLUDES:
• FULL-SCALE IMPLEMENTATION (K&R) C
COMPILER • THE STANDARD C LIBRARY • ROM
ROUTINES LIBRARY • LINKER • LIBRARIAN AND
DOCUMENTATION ...

\$299.95

FOR MORE INFORMATION OR TO ORDER CALL OR WRITE:
Megamax, Inc.
BOX 851521, DEPT. Y
RICHARDSON, TX
75085-1521
(214) 987-4931



MACINTOSH IS A
REGISTERED TRADEMARK
OF APPLE COMPUTER INC.

NOW
AVAILABLE
FOR THE
MACINTOSH

Circle no. 84 on reader service card.

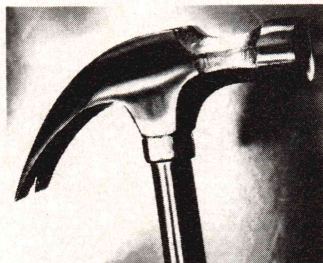
SCIENTIFIC/ENGINEERING PC GRAPHICS TOOLS

GRAFMATIC™ PLOTMATIC™

for FORTRAN/PASCAL PROGRAMMERS

The GRAFMATIC (screen graphics) and companion PLOTMATIC (pen plotter) libraries of modular scientific/engineering graphics routines let you easily create 2D and 3D plots in customized or default formats. Pen plot preview with GRAFMATIC. Plot interactively or in deferred mode. **Primitives** (mode, color, cursor, character, pixel, line, paint...) plus auto-scaling, auto-axis generation, auto-tic mark labeling, **function** plots, **tabular** plots, error bars, auto-function plots (complete plot in default format with one easy call), auto-tabular plots, **log/parametric/contour** plots, 3D rotation/scaling/translation, **wire frame** model (for old time's sake), **hidden line removal** for solid models (GRAFMATIC only), cubic and bicubic spline interpolations, least squares fits, bar and pie charts, screen dump... You name it. We have it! Best of all, the clearest and most complete documentation to be found in microcomputerland. User support? Of course, call us! We offer a no questions asked money-back guarantee.

SCREEN and PEN PLOTTER Software Libraries for the IBM PC Tandy 2000 TI Professional



MICROCOMPATIBLES

301 Prelude Dr.
Silver Spring, MD 20901
(301) 593-0683

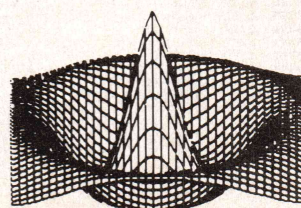
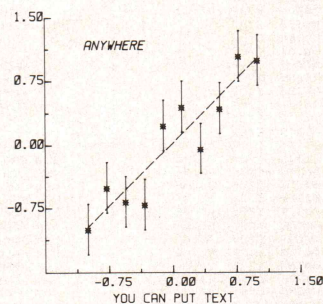
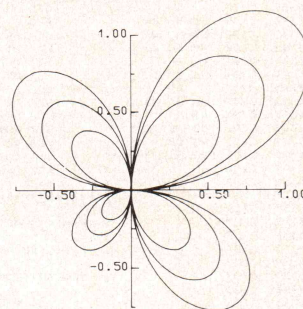
GRAFMATIC*	\$135
PLOTMATIC*†	\$135
BOTH	\$240
OMNILOT [S]	\$135
OMNILOT [P]†	\$135
BOTH	\$240

*Specify Compiler (Mfg. and Version): IBM/MS/IBM Prof. FORTRAN
†Specify Plotter: HP/HI/IBM

OMNILOT [S] OMNILOT [P] NO Programming Required

Integrated stand-alone graphics libraries to drive your CRT monitor or your pen plotter. Key in data in response to menu prompts or read your data from a disk file. Choose from an assortment of graphics formats: **tabular**, **line**, **bar** or **pie charts**. **Contour plots**. (YES! Just part of our integrated OMNILOT library, not an expensive individual item). Create 3-D plots with a choice of **wire frame** or **hidden surface** removal for added realism. Choose **standard**, **semi-log** or **log-log** scales; gridding; error bars; line colors and types; marker symbol colors and types. Cubic spline interpolations and least squares fitting options. As with our other professional packages we offer clear and careful documentation filled with examples, user support, and a no-questions asked money-back guarantee.

Ask for OMNILOT [S] for screen graphics and OMNILOT [P] for the pen plotter software library.



Circle no. 57 on reader service card.

Listing Four

```

00272 C726 AD 6E 03 LDA XSHD+1 ; RESTORE COORDS
00273 C729 BD 72 03 STA XSHD+1
00274 C72C 68 PLA
00275 C72D BD 6E 03 STA XSHD+1
00276 C730 AD 6D 03 LDA XSHD
00277 C733 BD 71 03 STA YSHD
00278 C736 68 PLA
00279 C737 BD 6D 03 STA XSHD
00280 C73A AD 6C 03 LDA XREL
00281 C73D BD 70 03 STA YREL
00282 C740 68 PLA
00283 C741 BD 6C 03 STA XREL
00284 C744 60 NORSTR RTS
00285 C745 ;
00286 C745 ; *****
00287 C745 ;
00288 C745 ; CALCULATE Z FROM LOCAL X,Y BY
00289 C745 ; PYTHAGOREAN SUM
00290 C745 ;
00291 C745 AD 77 03 GETZ LDA RADIUS
00292 C748 85 AC STA ARG
00293 C74A 20 04 C0 JSR SQUARE
00294 C74D BD 7C 03 STA TNTHP+1
00295 C750 A5 AE LDA SOR
00296 C752 BD 7B 03 STA TNTHP
00297 C755 AD 6D 03 LDA XSHD
00298 C758 85 AC STA ARG
00299 C75A 20 04 C0 JSR SQUARE
00300 C75D 38 SEC
00301 C75E AD 7B 03 LDA TNTHP
00302 C761 E5 AE SBC SOR
00303 C763 BD 7B 03 STA TNTHP
00304 C766 AD 7C 03 LDA TNTHP+1
00305 C769 E5 AF SBC SOR+1
00306 C76B BD 7C 03 STA TNTHP+1
00307 C76E AD 71 03 LDA YSHD
00308 C771 85 AC STA ARG
00309 C773 20 04 C0 JSR SQUARE
00310 C776 38 SEC
00311 C777 AD 7B 03 LDA TNTHP
00312 C77A E5 AE SBC SOR
00313 C77C 85 AC STA RADCN
00314 C77E AD 7C 03 LDA TNTHP+1
00315 C781 E5 AF SBC SOR+1
00316 C783 85 AD STA RADCN+1
00317 C785 30 0A BMI ZERODT
00318 C787 20 64 C0 JSR SORT
00319 C78A 0E 3C 03 ASL ROOT
00320 C78D 2E 3D 03 ROL ROOT+1
00321 C790 60 RTS
00322 C791 A9 00 ZERODT LDA #000
00323 C793 BD 3C 03 STA ROOT
00324 C796 BD 3D 03 STA ROOT+1
00325 C799 60 RTS
00326 C79A ;
00327 C79A ; *****
00328 C79A ;
00329 C79A ; SET UP PARAMETERS FOR TOROIDS
00330 C79A ;
00331 C79A ; RT=(RO-R1)/2 RS=RT*RT RC=RT+R1
00332 C79A ;
00333 C79A AD BE 03 TFARM LDA RO
00334 C79D 38 SEC
00335 C79E ED BF 03 SBC R1
00336 C7A1 4A LSR A
00337 C7A2 BD BC 03 STA RT
00338 C7A5 BD 77 03 STA RADIUS
00339 C7A8 18 CLC
00340 C7A9 6D BF 03 ADC R1
00341 C7AC BD BD 03 STA RC
00342 C7AF AD BC 03 LDA RT
00343 C7B2 85 AC STA ARG
00344 C7B4 20 04 C0 JSR SQUARE
00345 C7B7 A5 AE LDA SOR
00346 C7B9 BD BA 03 STA RS
00347 C7BC A5 AF LDA SOR+1
00348 C7BE BD BB 03 STA RS+1
00349 C7C1 A9 00 LDA #0
00350 C7C3 BD B6 03 STA CNTX
00351 C7C6 60 RTS
00352 C7C7 ;
00353 C7C7 ; *****
00354 C7C7 ;
00355 C7C7 ; DRAW A SHADED SPHERE
00356 C7C7 ;
00357 C7C7 ; 'BASIC SUBROUTINE' EQUIVALENT
00358 C7C7 ;
00359 C7C7 ; 'SPHERE' FOR CNTX=0 TO RADIUS/SOR(2)
00360 C7C7 ; XREL=CNTX*XSHD+1
00361 C7C7 ; FOR CNTX=CNTX TO SOR(RAD+RAD-CNTX*CNTX)
00362 C7C7 ; YREL=CNTX*YSHD+1
00363 C7C7 ; HVFLAG=0
00364 C7C7 ; GOSUB 'PTPLOT'
00365 C7C7 ; REM EXCHANGE X & Y TO USE B-FOLD SYM
00366 C7C7 ; HVFLAG=-128
00367 C7C7 ;
00368 C7C7 ; GOSUB 'PTPLOT'
00369 C7C7 ; NEXT CNTY
00370 C7C7 ; NEXT CNTX
00371 C7C7 ; RETURN
00372 C7C7 ;
00373 C7C7 ;
00374 C7C7 AD 77 03 SPHERE LDA RADIUS
00375 C7CA 85 AC STA ARG
00376 C7CC 20 04 C0 JSR SQUARE
00377 C7CF 8A AE ASL SOR
00378 C7D1 26 AF ROL SOR+1
00379 C7D3 A5 AE LDA SOR
00380 C7D5 85 AC STA RADCN
00381 C7D7 A5 AF LDA SOR+1
00382 C7D9 85 AD STA RADCN+1
00383 C7DB 20 64 C0 JSR SORT
00384 C7DE 4E 3D 03 ASL ROOT
00385 C7E1 6E 3C 03 ROR ROOT
00386 C7E4 AD 3C 03 LDA ROOT
00387 C7E7 BD 92 03 STA XMAX
00388 C7EA A9 00 LDA #000
00389 C7EC BD 86 03 STA CNTX
00390 C7EF BD 8E 03 STA XSHD+1
00391 C7F2 BD 72 03 STA YSHD+1
00392 C7F5 AD 77 03 LDA RADIUS
00393 C7F8 85 AC STA ARG
00394 C7FA 20 04 C0 JSR SQUARE
00395 C7FD BD 85 03 STA TEMP+1
00396 C800 A5 AE LDA SOR
00397 C802 BD 84 03 STA TEMP
00398 C805 AD 86 03 LDA CNTX
00399 C808 BD 87 03 STA CNTY
00400 C80B 85 AC STA ARG
00401 C80D BD 6C 03 STA XREL
00402 C810 BD 6D 03 STA XSHD
00403 C813 20 04 C0 JSR SQUARE
00404 C816 38 SEC
00405 C817 AD 84 03 LDA TEMP
00406 C81A E5 AE SBC SOR
00407 C81C 85 AC STA RADCN
00408 C81E AD 85 03 LDA TEMP+1
00409 C821 E5 AF SBC SOR+1
00410 C823 BD AD 03 STA RADCN+1
00411 C825 20 64 C0 JSR SORT
00412 C828 AD 3C 03 LDA ROOT
00413 C82B BD 88 03 STA MAX
00414 C82E AD 87 03 LDA CNTY
00415 C831 BD 70 03 STA YREL
00416 C834 BD 71 03 STA YSHD
00417 C837 A9 00 LDA #0
00418 C839 BD 83 03 STA HVFLAG
00419 C83C 20 28 C6 JSR PTFLOT
00420 C83F A9 80 LDA #400
00421 C841 BD 83 03 STA HVFLAG
00422 C844 20 28 C6 JSR PTFLOT
00423 C847 AD 87 03 LDA CNTY
00424 C84A CD 88 03 CMP MAX
00425 C84D F0 06 BEQ DONEY
00426 C84F E5 87 03 INC CNTY
00427 C852 4C 2E C8 JMP LOOPY
00428 C855 AD 86 03 DONEY LDA CNTX
00429 C858 CD 92 03 CMP XMAX
00430 C85B F0 06 BEQ DONE
00431 C85D EE 86 03 INC CNTX
00432 C860 4C 2E C8 JMP LOOPY
00433 C863 60 DONE RTS
00434 C864 ;
00435 C864 ; *****
00436 C864 ;
00437 C864 ; DRAW SHADED CYLINDERS
00438 C864 ;
00439 C864 ; 'BASIC SUBROUTINE' EQUIVALENT
00440 C864 ;
00441 C864 ; 'CYLNDR' XSHD=0
00442 C864 ; FOR YREL=RADIUS TO 0
00443 C864 ; YSHD=YREL
00444 C864 ; FOR XREL=XLEN TO 0
00445 C864 ; GOSUB 'PTPLOT'
00446 C864 ; NEXT XREL
00447 C864 ; NEXT YREL
00448 C864 ; RETURN
00449 C864 ;
00450 C864 A9 00 CYLNDR LDA #0
00451 C866 BD AD 03 STA XSHD
00452 C869 BD 6E 03 STA XSHD+1
00453 C86C BD 72 03 STA YSHD+1
00454 C86F AD 77 03 LDA RADIUS
00455 C872 BD 70 03 STA YREL
00456 C875 AD 89 03 CYLOOP LDA HLEN
00457 C878 BD 6C 03 STA XREL
00458 C87B AD 70 03 LDA YREL
00459 C87E BD 71 03 STA YSHD
00460 C881 20 28 C6 CXLOOP JSR PTFLOT
00461 C884 CE 6C 03 DEC XREL
00462 C887 10 F8 BPL CXLOOP
00463 C889 CE 70 03 DEC YREL
00464 C88C 10 E7 BPL CYLOOP
00465 C88E 60 RTS
00466 C88F ;
00467 C88F ; *****
00468 C88F ;
00469 C88F ; DRAW EDGE-VIEW TOROIDS
00470 C88F ;
00471 C88F ; 'BASIC SUBROUTINE' EQUIVALENT
00472 C88F ;
00473 C88F ; 'EDGTOR' GOSUB 'TFARM';REM SET UP RADII
00474 C88F ; FOR CNTX=0 TO RT
00475 C88F ; XREL=CNTX*XSHD+1
00476 C88F ; R0=SQR(RT*RT-CNTX*CNTX)
00477 C88F ; FOR CNTY=0 TO R0+RC
00478 C88F ; YREL=CNTY
00479 C88F ; YSHD=(R0+CNTY)/(R0+RC)
00480 C88F ; GOSUB 'PTPLOT'
00481 C88F ; NEXT CNTY
00482 C88F ; NEXT CNTX
00483 C88F ; RETURN
00484 C88F ;
00485 C88F 20 9A C7 EDGTOR JSR TFARM
00486 C892 A9 00 LDA #100
00487 C894 BD 6E 03 STA XSHD+1
00488 C897 BD 72 03 STA YSHD+1
00489 C89A AD 86 03 LOOPX4 LDA CNTX
00490 C89D BD 6C 03 STA XREL
00491 C8A0 BD 6D 03 STA XSHD
00492 C8A3 85 AC STA ARG
00493 C8A5 20 04 C0 JSR SQUARE
00494 C8A8 38 SEC
00495 C8A9 AD BA 03 LDA RS
00496 C8AC E5 AE SBC SOR
00497 C8AE 85 AC STA RADCN
00498 C8B0 AD 8B 03 LDA RS+1
00499 C8B3 E5 AF SBC SOR+1
00500 C8B5 BD AD 03 STA RADCN+1
00501 C8B7 20 64 C0 JSR SORT
00502 C8BA AD 3C 03 LDA ROOT
00503 C8BD BD 89 03 STA R0
00504 C8C0 18 CLC
00505 C8C1 6D BD 03 ADC RC
00506 C8C4 BD 88 03 STA MAX
00507 C8C7 A9 00 LDA #000
00508 C8C9 BD 87 03 STA CNTY
00509 C8CB AD 87 03 LOOPY4 LDA CNTY
00510 C8CF BD 70 03 STA YREL
00511 C8D2 BD AD 03 STA MLFLER
00512 C8D4 AD 89 03 LDA R0
00513 C8D7 85 AC STA MLFCND
00514 C8D9 20 11 C8 JSR MULT
00515 C8DC 85 FE STA DVDCND+1
00516 C8DE A5 AE LDA PROD
00517 C8E0 85 FD STA DVDCND
00518 C8E2 AD 88 03 LDA MAX
00519 C8E5 85 FB STA DVSOR
00520 C8E7 20 EA C5 JSR SDIV
00521 C8EA A5 FD LDA QUOT
00522 C8EC BD 71 03 STA YSHD
00523 C8EF 20 28 C6 JSR PTFLOT
00524 C8F2 AD 87 03 LDA CNTY
00525 C8F5 CD 88 03 CMP MAX
00526 C8F8 4C 2E C8 BEQ DONE4
00527 C8FA EE 87 03 INC CNTY

```

(Continued on page 74)

EVER WISH THAT YOU
HAD A PLOTTER? TRY

H PLOT

A PLOTTER EMULATION PROGRAM
THAT CREATES GRAPHICS ON YOUR
DOT MATRIX PRINTER. FEATURES:

- * POWERFUL HP-GL™ SYNTAX FOUND ON HEWLETT-PACKARD PLOTTERS.
- * FAST! CREATES & PRINTS A TYPICAL GRAPH IN LESS THAN 4 MINUTES.
- * IMAGES STORED IN MINIMUM SPACE ON DISK, AND MAY BE OVERLAID.
- * LABELS ANY SIZE, SLANT, OR ROTATED.
- * SCALING, LINETYPES, WINDOWS, TICKS, AND SYMBOL MODE.
- * PLOT SIZES 11"x14" TO 4"x48"!
- * INCLUDES MANUAL AND USEFUL EXAMPLE PROGRAMS.
- * REQUIRES 54K CP/M 2.2 AND AN OKIDATA, EPSON, GEMINI, OR PROWRITER PRINTER.

\$49.95 PPD. OH RES ADD 5% TAX

SPECIFY DISK FORMAT, PRINTER MODEL.

ORDINATE SOLUTIONS

P.O. BOX 308, OBERLIN, OH 44074

WRITE CONCERNING OTHER PRINTERS, OS's.

Circle no. 59 on reader service card.

GRAPHIC DESIGN



We specialize in high tech advertising, product packaging design, technical and whimsical illustration, marketing brochures, software & hardware manuals, logos, fliers, posters.

Contact us now and let us show you how we can improve your graphic communication.

In the heart of Silicon Valley
M'GUINNESS DESIGN
(415) 967-3811

1122 Golden Way, Los Altos, CA 94022



Circle no. 24 on reader service card.

Classy Chassis

3315
5" Floppy/Winchester
7 Cards \$417*

3310
5" Floppy/Winchester
4 Cards \$387*

3002T
5" Floppy/Winchester
10 Cards \$565*

3307
8" Floppy/5" Winchester
7 Cards \$494*

laser 3000

MAIN/FRAMES & DISC ENCLOSURES FROM \$100

LASER 3000 DISC/COVERS (not shown)

* 1 piece; prices lower in quantity.

3916F 5" Floppy \$100* **3915** 2 ea. 5" Winchester \$199*

INTEGRAND

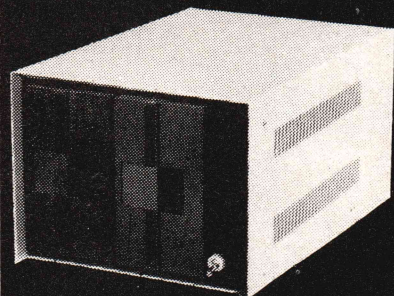
RESEARCH CORPORATION

(Disk drives not included)

8620 Roosevelt Ave./Visalia, CA 93291 209/651-1203

Circle no. 15 on reader service card.

AMPRO
"Little Board"
MAIN/FRAMES
6 Models from **\$125***



\$150 (1 piece*)
MODEL 2800
Includes power supply & fan
(Disk Drives and Little Board not included)
AMPRO & Little Board are TM AMPRO computers.

INTEGRAND

RESEARCH CORPORATION

8620 Roosevelt Ave./Visalia, CA 93291

209/651-1203

Drawing on the C-64

(Listing Continued, text begins on page 50)

Listing Four

```

00528 C9FD 4C CC C8      JMP LOOPY4
00529 C900 AD B6 03      DONE4 LDA CNTX
00530 C903 CD BC 03      CMP RT
00531 C906 F0 B6 00      B60 DONEHT
00532 C908 EE B6 03      INC CNTX
00533 C90B 4C 9A C8      JMP LOOPX4
00534 C90E 60          DONEHT RTS
00535 C90F          ;
00536 C90F          ; *****
00537 C90F          ;
00538 C90F          ; DRAW A SHADED, TOP-VIEW TOROID
00539 C90F          ;
00540 C90F          ; 'BASIC SUBROUTINE' EQUIVALENT
00541 C90F          ;
00542 C90F          ; 'TOROID' GOSUB 'TPARM'
00543 C90F          ; FOR CNTX=0 TO RD/SOR(2)
00544 C90F          ; REM 6-FOLD SYMMETRY USED
00545 C90F          ; XREL=CNTX
00546 C90F          ; MAX=SQR(RD*RD-CNTX*CNTX)
00547 C90F          ; IF CNTX>RI THEN GOTO 'GRTR'
00548 C90F          ; CNTY=SQR(RI*RI-CNTX*CNTX)
00549 C90F          ; GOTO 'LLPY1'
00550 C90F          ; 'GRTR' CNTY=CNTY+1
00551 C90F          ; 'LLPY1' YREL=CNTY
00552 C90F          ; RD=SQR(CNTY*CNTY+CNTX*CNTX)
00553 C90F          ; XSHD=CNTX-(CNTX*RC)/RD
00554 C90F          ; YSHD=CNTY-(CNTY*RC)/RD
00555 C90F          ; HVFLAG=0:GOSUB 'PTPLOT'
00556 C90F          ; HVFLAG=-128:GOSUB 'PTPLOT'
00557 C90F          ; IF CNTY=MAX THEN GOTO 'DDNY1'
00558 C90F          ; CNTX=CNTX+1
00559 C90F          ; GOTO 'LLPY1'
00560 C90F          ; 'DDNY1' NEXT CNTX
00561 C90F          ; RETURN
00562 C90F          ;
00563 C90F          ; TOROID JSR TPARM
00564 C912 AD BE 03      LDA RD
00565 C915 85 AC          STA ARG
00566 C917 20 04 C0      JSR SQUARE
00567 C91A 06 AE          ASL SOR
00568 C91C 26 AF          ROL SOR+1
00569 C91E A5 AE          LDA SOR
00570 C920 85 AC          STA RADCN
00571 C922 A5 AF          LDA SOR+1
00572 C924 05 AD          STA RADCN+1
00573 C926 20 64 C0      JSR SORT
00574 C929 4E 3D 03      LSR ROOT+1
00575 C92C 6E 3C 03      ROR ROOT
00576 C92F AD 3C 03      LDA ROOT
00577 C932 8D 92 03      STA XMAX
00578 C935 AD B6 03      LDA SOR+1
00579 C938 8D 6C 03      STA XREL
00580 C93B 85 AC          STA ARG
00581 C93D 20 04 C0      JSR SQUARE
00582 C940 8D 91 03      STA XSOR+1
00583 C943 A5 AE          LDA SOR
00584 C945 8D 92 03      STA XSOR
00585 C948 AD BE 03      LDA RD
00586 C94B 85 AC          STA ARG
00587 C94D 20 04 C0      JSR SQUARE
00588 C950 38          SEC
00589 C951 A5 AE          LDA SOR
00590 C953 ED 90 03      SBC XSOR
00591 C956 85 AC          STA RADCN
00592 C958 A5 AF          LDA SOR+1
00593 C95A ED 91 03      SBC XSOR+1
00594 C95D 85 AD          STA RADCN+1
00595 C95F 20 64 C0      JSR SORT
00596 C962 AD 3C 03      LDA ROOT
00597 C965 8D 9B 03      STA MAX
00598 C968 38          SEC
00599 C969 AD BF 03      LDA R1
00600 C96C ED B6 03      SBC CNTX
00601 C96F 90 23          BCC GRTR
00602 C971 AD BF 03      LDA R1
00603 C974 85 AC          STA ARG
00604 C976 20 04 C0      JSR SQUARE
00605 C979 38          SEC
00606 C97A A5 AE          LDA SOR
00607 C97C ED 90 03      SBC XSOR
00608 C97F 85 AC          STA RADCN
00609 C981 A5 AF          LDA SOR+1
00610 C983 ED 91 03      SBC XSOR+1
00611 C986 85 AD          STA RADCN+1
00612 C988 20 64 C0      JSR SORT
00613 C98B AD 3C 03      LDA ROOT
00614 C98E 8D 87 03      STA CNTY
00615 C991 4C 9A C9      JMP LLPY1
00616 C994 AD B6 03      LDA CNTX
00617 C997 8D 87 03      STA CNTY
00618 C99A AD 87 03      LDA CNTY
00619 C99D 8D 70 03      STA YREL
00620 C9A0 85 AC          STA ARG
00621 C9A2 20 04 C0      JSR SQUARE
00622 C9A5 1B          CLC
00623 C9A6 A5 AE          LDA SOR
00624 C9A8 6D 90 03      ADC XSOR
00625 C9AB 85 AC          STA RADCN
00626 C9AD A5 AF          LDA SOR+1
00627 C9AF 6D 91 03      ADC XSOR+1
00628 C9B2 85 AD          STA RADCN+1
00629 C9B4 20 64 C0      JSR SORT
00630 C9B7 AD 3C 03      LDA ROOT
00631 C9BA 8D 89 03      STA RD
00632 C9BD 85 FB          LDA DVSOR
00633 C9BF AD B6 03      LDA CNTX
00634 C9C2 85 AD          STA MLFLER
00635 C9C4 AD 8D 03      LDA RC
00636 C9C7 85 AC          STA MLPCND
00637 C9C9 20 11 C0      JSR MULT
00638 C9CC 85 FE          LDA DVND+1
00639 C9CE A5 AE          LDA PROD
00640 C9D0 85 FD          STA DVND
00641 C9D2 20 EA C5      JSR SDIV
00642 C9D5 38          SEC
00643 C9D6 AD B6 03      LDA CNTX
00644 C9D9 E5 FD          DIVIDE
00645 C9DB 8D 60 03      STA XSHD
00646 C9DE A9 00          LDA #000
00647 C9E0 E5 FE          SBC QUOT+1
00648 C9E2 8D 6E 03      STA XSHD+1
00649 C9E5 AD 87 03      LDA CNTY
00650 C9E8 85 AD          STA MLFLER
00651 C9EA AD 8D 03      LDA RC
00652 C9ED 85 AC          STA MLPCND
00653 C9EF 20 11 C0      JSR MULT
00654 C9F2 85 FE          STA DVND+1
00655 C9F4 A5 AE          LDA PROD
00656 C9F6 85 FD          STA DVND

```

```

00657 C9F8 AD 89 03      LDA RD
00658 C9FB 85 FB          STA DVSOR
00659 C9FD 20 EA C5      JSR SDIV
00660 CA00 38          SEC
00661 CA01 AD 87 03      LDA CNTY
00662 CA04 E5 FD          SBC QUOT
00663 CA06 8D 71 03      STA YSHD
00664 CA09 A9 00          LDA #000
00665 CA0B 8D 83 03      STA HVFLAG
00666 CA0E E5 FE          SBC QUOT+1
00667 CA10 8D 72 03      STA YSHD+1
00668 CA13 20 28 C6      JSR PTPLOT
00669 CA16 A9 80          LDA #080
00670 CA18 8D 83 03      STA HVFLAG
00671 CA1B 20 28 C6      JSR PTPLOT
00672 CA1E AD 87 03      LDA CNTY
00673 CA21 CD 88 03      CMP MAX
00674 CA24 F0 06          BEQ DDNY1
00675 CA26 EE 87 03      INC CNTY
00676 CA29 4C 9A C9      JMP LLPY1
00677 CA2C AD 86 03      LDA CNTX
00678 CA2F CD 92 03      CMP XMAX
00679 CA32 F0 06          BEQ DUNTOR
00680 CA34 EE 86 03      INC CNTX
00681 CA37 4C 35 C9      JMP LLPX1
00682 CA3A 60          DUNTOR RTS
00683 CA3B          ;
00684 CA3B          ; *****
00685 CA3B          ;
00686 CA3B          ; DRAW "INSIDE VIEW" TOROIDS
00687 CA3B          ;
00688 CA3B          ; 'BASIC SUBROUTINE' EQUIVALENT
00689 CA3B          ;
00690 CA3B          ; 'SPOOL' GOSUB 'TPARM'
00691 CA3B          ; FOR CNTX=0 TO RT
00692 CA3B          ; XREL=CNTX: XSHD=CNTX
00693 CA3B          ; MAX=RC-SQR(RS-CNTX*CNTX)
00694 CA3B          ; FOR CNTY=0 TO MAX
00695 CA3B          ; YREL=CNTY
00696 CA3B          ; YSHD=(RC-CNTY/MAX)-CNTY
00697 CA3B          ; GOSUB 'PTPLOT'
00698 CA3B          ; NEXT CNTY
00699 CA3B          ; NEXT CNTX
00700 CA3B          ; RETURN
00701 CA3B          ;
00702 CA3B 20 9A C7      SPOOL JSR TPARM
00703 CA3E AD 86 03      LLPX2 LDA CNTX
00704 CA41 8D 6C 03      STA XREL
00705 CA44 85 AC          STA ARG
00706 CA46 38          SEC
00707 CA47 A9 00          LDA #000
00708 CA49 ED 86 03      SBC CNTX
00709 CA4C 8D 6D 03      STA XSHD
00710 CA4F A9 00          LDA #000
00711 CA51 E9 00          SBC #000
00712 CA53 8D 6E 03      STA XSHD+1
00713 CA56 20 04 C0      JSR SQUARE
00714 CA59 38          SEC
00715 CA5A AD 8A 03      LDA RS
00716 CA5D E5 AE          SBC SOR
00717 CA5F 85 AC          STA RADCN
00718 CA61 AD 8B 03      LDA RS+1
00719 CA64 E5 AF          SBC SOR+1
00720 CA66 85 AD          STA RADCN+1
00721 CA68 20 64 C0      JSR SORT
00722 CA6B 38          SEC
00723 CA6C AD 8D 03      LDA RC
00724 CA6F ED 3C 03      SBC ROOT
00725 CA72 8D 8B 03      STA MAX
00726 CA75 A9 00          LDA #000
00727 CA77 8D 87 03      STA CNTY
00728 CA7A AD 87 03      LDA CNTY
00729 CA7D 8D 70 03      STA YREL
00730 CA80 85 AD          STA MLFLER
00731 CA82 AD 8D 03      LDA RC
00732 CA85 85 AC          STA MLPCND
00733 CA87 20 11 C0      JSR MULT
00734 CA8A 85 FE          STA DVND+1
00735 CA8C A5 AE          LDA PROD
00736 CA8E 85 FD          STA DVND
00737 CA90 AD 8B 03      LDA MAX
00738 CA93 85 FB          STA DVSOR
00739 CA95 20 EA C5      JSR SDIV
00740 CA98 A5 FD          LDA QUOT
00741 CA9A 38          SEC
00742 CA9B ED 87 03      SBC CNTY
00743 CA9E 8D 71 03      STA YSHD
00744 CAA1 A5 FE          LDA QUOT+1
00745 CAA3 E9 00          SBC #000
00746 CAA5 8D 72 03      STA YSHD+1
00747 CAAB 20 28 C6      JSR PTPLOT
00748 CAAD AD 87 03      LDA CNTY
00749 CAAE CD 88 03      CMP MAX
00750 CAB1 F0 06          BEQ DDNY2
00751 CAB3 EE 87 03      INC CNTY
00752 CAB6 4C 7A CA      JMP LLPY2
00753 CAB9 AD 86 03      LDA CNTX
00754 CABE CD 8C 03      CMP RT
00755 CABF F0 06          BEQ DUNHSP
00756 CAC1 EE 86 03      INC CNTX
00757 CAC4 4C 3E CA      JMP LLPX2
00758 CAC7 60          DUNHSP RTS
00759 CACB          .END

```

ERRORS = 00000

SYMBOL TABLE

SYMBOL VALUE

ARG 00AC BAKLIT 037D

CLIP1 037D CLIFR 037E

SYMBOL TABLE

SYMBOL VALUE

CNTX 0386 CNTY 0387

CYLOOP 0375 DDNY1 CA2C

DIVIDE 0325 DONE CA83

DONE1 C855 DUNHSP CAC7

DVSOR 00FB EDGTR C8BF

GRTR C994 HEHT 0381

HVFLAG 85B3 LLPX1 C935

LLPY2 CA7A LOPX C8B5

LOOPY4 C8CC MAX 038B

MULT C011 NEGATE C5FF

CYLOOP 0386

DDNY1 CA2C

DONE CA83

DUNHSP CAC7

EDGTR C8BF

HEHT 0381

LLPX1 C935

LOPX C8B5

MAX 038B

NEGATE C5FF

CYLOOP 0386

DDNY1 CA2C

DONE CA83

DUNTOR CA5A

GETVAL C5F1

HTORRN 0346

LLPY1 C99A

LOOPY C89A

MLPCND 00AC

MLFLER 00AD

NOROT C65A

End Listing Four

(Listing Five begins on page 76)

BIG DISCOUNTS FROM JOHN D. OWENS ASSOCIATES

MACROTECH MI-286: 80286/Z-8011 DUAL PROCESSOR S-100 CPU BOARD: \$1,116

MACROTECH MEMORY MSR: 120NS, high-speed dynamic RAM with realistic pricing:

Works with CompuPro 8085/8088; MT-286 and others:
256K: \$556 512K: \$876

MACROTECH STATIC RAM: Substitute for RAM 22 and RAM 23.
256K STATIC: \$960 512K STATIC: \$1,800

EMERALD SYSTEMS HARD DISK SUBSYSTEMS and TAPE BACKUP

High capacity! Up to 280 MB! Emerald has overcome the 32MB DOS limitation! Allows multiple volumes per physical drive. Back up and restore utilities. Ideal for LAN applications.

HOUSTON INSTRUMENTS: Plotters: DMP 41 OR 42: \$2,397; DMP 29: \$1,838

DIGITIZERS: DT111 \$694; DT114 \$750; DT11AA \$714
New! 14 pen DMP 51 and 52: \$4,796

ILLUMINATED TECHNOLOGY S-100 COLOR GRAPHICS: \$1,116

NEC APC III: 80186 MS-DOS system w/spectacular graphics: 20% off list

SEMIDISK 2MB DISK EMULATOR FOR IBM PC and EPSON QX 10: \$2,050

BIG DISCOUNTS on LOMAS, COMPUPRO, IMS, INTERCONTINENTAL, DIGITAL GRAPHICS SYSTEMS, ADVANCED DIGITAL, ACKERMAN, BYAD and many others.
Prices & availability subject to change without notice.

Write or call for product literature and inventory sale list.

WE EXPORT: OVERSEAS CALLERS: TWX 710 588 2844
(OWENSASSOC NYK)

DOMESTIC AND OVERSEAS ORDERS TAKEN BY PHONE,
LETTER, TELEX OR EASY LINK.
We accept VISA and Mastercard. Shipping \$5 per board in continental USA.

JOHN D. OWENS ASSOCIATES
12 SCHUBERT STREET STATEN ISLAND, NEW YORK 10305
(718) 448 6283 (718) 448 6298 (718) 448 2913
EASY LINK MAILBOX ADDRESS: 63840768

LATTICE® WORKS

LATTICE UNVEILS FOUR PRODUCTS

Lattice has announced the availability of four new software products for MS-DOS environments:

C-SPRITE is a software tool that simplifies debugging of programs written in Lattice C or assembly language. Cost: \$175 per copy.

LMK is an Automated Product Generation Utility (UNIX "MAKE") that enhances productivity and relieves the tedium of rebuilding complex software systems or documents. Cost: \$195 per copy.

The **TEXT MANAGEMENT UTILITY PACKAGE** includes utilities to search a set of files for simple or complicated patterns, to see the exact minimal differences between two text files, and to modify one or more text files automatically. Cost: \$120 per copy.

CVUE is a full screen text editor that supports all normal screen editor functions and includes a configuration program to define tabstop positions, horizontal scrolling and edit commands. Cost: \$100 per copy.

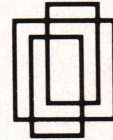
For complete information on these new products, contact Lattice.

LATTICE C NAMED 'BEST OF 1984'

The Lattice C compiler has been rated 'Best of 1984' by PC Magazine. According to columnist Peter Norton, "The Lattice C compiler is quite good . . . and in my opinion noticeably better than any of its competitors. Lattice C generates code that is quite compact and fast running; the closest competitor in my tests generated code that was about 10 to 15 percent bulkier."

ASK ABOUT OUR "TRADE UP TO LATTICE C POLICY"

After purchase, return registration cards for free subscription to the "Lattice Works" newsletter and important information about the Lattice Users Group.



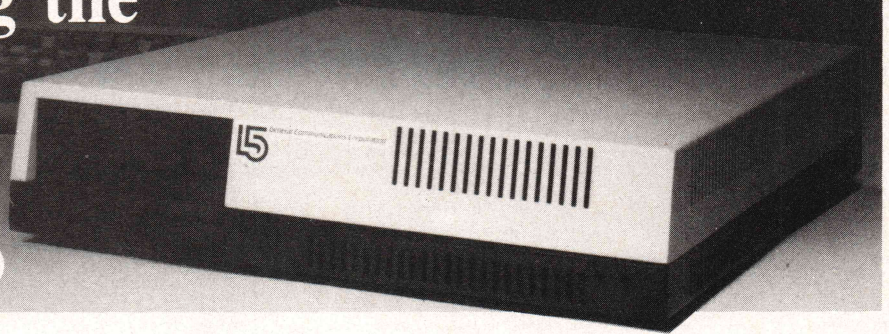
Lattice, Inc.
P.O. Box 3072
Glen Ellyn, IL 60138
(312) 858-7950
TWX 910-291-2190

International Sales Offices

Belgium: Softshop. Phone: (32) 53-664875.
England: Round Hills. Phone: (0672) 54675.
Japan: Lifeboat Japan. Phone: (03) 293-2311.

Circle no. 58 on reader service card.

Introducing the L5 supermicro



A breakthrough in Price, Performance and Packaging

The new L5 proves that good things come in small packages! Measuring a compact 3.75"H x 17.5"W x 21"D, the L5 is small enough to fit on a desk-top or a laboratory workbench. Yet it's large enough to handle up to 32 users and, in some applications, outperform a VAX system. The L5 is small in price, too. A mid-range system can cost less than \$1,000 per user!

Find out how the new L5 offers the unique solution to price, performance & placement challenges. Call General Communications today!



General Communications Corporation

"Where lucid communications, technical excellence and common sense meet."

1 Main Street, Suite 502, Eatontown, NJ 07724 (201) 542-6560

L5 Features

- KDJ11 Processor, floating point, 8K cache
- .5Mb to 32Mb memory
- 4 to 32 users
- 20Mb to 2.5Gb external storage
- UNIX System V Fast Kernel or Real Time Kernel
- Runs RT11, RSX, and TSX
- Includes several utilities packages, necessary cabling, complete documentation and tutorials

Aggressive Dealer/OEM discounts available

*UNIX is a trademark of AT&T-Bell Laboratories
VAX, RT11, & RSX are trademarks of Digital Equipment Corporation
TSX is a trademark of S&H Computer

Circle no. 13 on reader service card.

Listing Four

```
NORSTR C744 NOSCAL 0347 ORIGIN C5EA PLDONE C721
PLTSHD C20F PROD 00AE PTPLOT C628 PTPLOT2 C65D
QUOT 00FD R0 0389 RADPCND 00AC RADIUS 0377
RAM 036A RANDOM C0CB RC 038D RHEM1 C6E8
R1 038F RNDM C000 RO 038E ROOT 033C
RS 038A RT 038C SDIV C5EA SPHERE C7C7
SPOOL CA3B SOR 00AE SORT C064 SQUARE C004
TEMP 0384 TNTHP 037B TONE 0379 TOROID C90F
TPARM C79A VALUE 0344 XCENT 036A XMAX 0392
XPLT 033F XREL 034C XSHD 036D XSOR 0390
YCENT 036F YPLT 0341 YREL 0370 YSHD 0371
ZEROOT C791 ZREL 0373 ZWX 0375
```

END OF ASSEMBLY

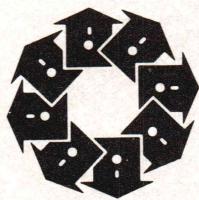
Listing Five

```
00001 0000 ; INTERFACE - EASY PARAMETER SETTING FOR SHAPE
00002 0000 ; DRAWING ROUTINES FROM BASIC.
00003 0000 ;
00004 0000 ; RICHARD L. RYLANDER 11/23/84
00005 0000 ;
00006 0000 ; *****
00007 0000 ORIGIN=#CACB *****
00008 0000 RAM #A0393
00009 0000 ;
00010 0000 ; PARAMETER LOCATIONS FOR VARIOUS SHAPES
00011 0000 ;
00012 0000 XCENT #A036A
00013 0000 YCENT #A036F
00014 0000 XPLT #A033F
00015 0000 YPLT #A0341
00016 0000 XMIN #A034A
00017 0000 YMIN #A034C
00018 0000 XMD #A034D
00019 0000 YMD #A034F
00020 0000 XMAX #A0350
00021 0000 YMAX #A0352
00022 0000 RADIUS #A0377
00023 0000 HLEN #A03B9
00024 0000 RI #A03BF
00025 0000 RU #A03BE
00026 0000 ;
00027 0000 HVFLAG #A03B3
00028 0000 VALUE #A0344
00029 0000 PLTFLG #A033E
00030 0000 ;
00031 0000 DEFLAG #FB
00032 0000 ;
00033 0000 ; *****
00034 0000 ; FUNCTION LOCATIONS
00035 0000 ;
00036 0000 ;
00037 0000 GRFON #C0E2 ; SWITCH TO GRAPHICS MODE
00038 0000 GRFOFF #C103 ; RETURN TO TEXT DISPLAY
00039 0000 ;
00040 0000 CLEAR=#C12C ; CLEAR BITMAP
00041 0000 CLRBYT=#C135 ; CLEAR (FILL) BYTE
00042 0000 COLOR=#C118 ; LOAD COLOR MAP
00043 0000 COLBYT=#C119 ; COLOR BYTE
00044 0000 ;
00045 0000 PLOT #C14B ; POINT PLOT ROUTINE
00046 0000 LINER #C2DB ; DRAW A LINE
00047 0000 FACETR #C4E1 ; DRAW A SHADED FACET
00048 0000 ;
00049 0000 ; *****
00050 0000 ; SHADED SHAPE DRAWING ROUTINES
00051 0000 ;
00052 0000 ;
00053 0000 SPHERR=#C7C7 ; SPHERE
00054 0000 CYLNDR=#C864 ; CYLINDER
00055 0000 TORUSR=#C90F ; TOP-VIEW TOROID
00056 0000 EDGTOR=#C8BF ; EDGE-VIEW TOROID
00057 0000 SPOULR=#CA3B ; INSIDE-VIEW TOROID
00058 0000 ;
00059 0000 ; *****
00060 0000 ; BASIC ROM ROUTINES
00061 0000 ;
00062 0000 CHKCOM=#A0FD ; CHECK FOR COMMA
00063 0000 EVAEXP=#AD9E ; EVALUATE EXPRESSION
00064 0000 FLTFLX=#B1AA ; CONVERT TO FIXED
00065 0000 ;
00066 0000 **RAM
00067 0000 LINFAC #***1 ; LINE OR FACET FLAG
00068 0393 ;
00069 0394 **ORIGIN
00070 0394 ;
00071 CACB ;
00072 CACB ; *****
00073 CACB ;
00074 CACB ; GET PARAMETERS FROM BASIC CALLING STATEMENT
00075 CACB ; OF THE FORM:
00076 CACB ; SYS(FUNCT),PARAM1,PARAM2,PARAMS(OPT)
00077 CACB ; WHERE THE THIRD PARAMETER (FOR EXAMPLE)
00078 CACB ; MAY BE OPTIONAL (A DEFAULT VALUE IS USED
00079 CACB ; IF THE PARAMETER IS NOT SPECIFIED)
00080 CACB ;
00081 CACB 20 FD AE ; GETNUM JSR CHKCOM ; LOOK FOR COMMA
00082 CACB 20 FE AD ; JSR EVAEXP ; EVALUATE EXPRESSION
00083 CACB 20 AA B1 ; JSR FLTFLX ; CONVERT TO INTEGER WITH
00084 CAD1 ; HIGH BYTE IN "A" AND LOW BYTE IN "Y"
00085 CAD1 60 RTS
00086 CAD2 ;
00087 CAD2 ; CHECK FOR ADDITIONAL (OPTIONAL) PARAMETERS
00088 CAD2 ;
00089 CAD2 A9 2C PCHCK LDA #32C ; ", " COMMA
00090 CAD4 A0 00 LDY #0
00091 CAD6 B4 FB STY DEFLAG
00092 CAD8 D1 7A CMP (#7A),Y
00093 CAD8 D0 03 BNE NOMORE ; NO COMMA - USE DEFAULT
00094 CAD8 D0 03 STY RI
00095 CADF A0 B0 NOMORE LDY #B0
00096 CAE1 B4 FB STY DEFLAG
00097 CAE3 60 RTS
00098 CAE4 ;
00099 CAE4 ; GET TWO ADDITIONAL PARAMETERS FOR TOROIDS
00100 CAE4 ;
00101 CAE4 20 D2 CA GETTWO JSR PCHCK
00102 CAE7 24 FB BIT DEFLAG
00103 CAE9 30 0F BMI DFAULT
00104 CAEB 20 9E AD JSR EVAEXP
00105 CAEE 20 AA B1 JSR FLTFLX
00106 CAF1 8C 9F 03 STY R1
00107 CAF4 20 CB CA JSR GETNUM
00108 CAF7 8C BE 03 STY RO
00109 CAF8 60 DFAULT RTS
00110 CAF8 ;
```

```
00111 CAFB ; *****
00112 CAFB ;
00113 CAFB ; SET CENTER COORDINATES
00114 CAFB ;
00115 CAFB 20 CB CA CENTER JSR GETNUM
00116 CAFE 8C 6A 03 STY XCENT
00117 CB01 8D 6B 03 STA XCENT+1
00118 CB04 20 CB CA JSR GETNUM
00119 CB07 8C 6F 03 STY YCENT
00120 CB0A 60 RTS
00121 CB0B ;
00122 CB0B ; *****
00123 CB0B ;
00124 CB0B ; CLEAR THE BITMAP, FILLING WITH (OPTIONAL)
00125 CB0B ; FILL VALUE SPECIFIED OR WITH (DEFAULT) "0"
00126 CB0B ;
00127 CB0B 20 D2 CA CLEAR2 JSR PCHCK
00128 CB0E 24 FB BIT DEFLAG
00129 CB10 30 07 BMI DEFCLR
00130 CB12 20 9E AD JSR EVAEXP
00131 CB15 20 AA B1 JSR FLTFLX
00132 CB18 2C .BYTE #2C
00133 CB19 A0 B0 DEFCLR LDY #0
00134 CB1B 8C 35 C1 STY CLRBYT
00135 CB1E 4C 2C C1 JMP CLEARR
00136 CB21 ;
00137 CB21 ; *****
00138 CB21 ;
00139 CB21 ; FILL COLOR MAP WITH (OPTIONAL) COLOR BYTE
00140 CB21 ; SPECIFIED OR WITH (DEFAULT) "#01"
00141 CB21 ; (BLACK DOTS ON WHITE BACKGROUND)
00142 CB21 ;
00143 CB21 20 D2 CA COLOR2 JSR PCHCK
00144 CB24 24 FB BIT DEFLAG
00145 CB26 30 07 BMI DEFCLR
00146 CB28 20 9E AD JSR EVAEXP
00147 CB2B 20 AA B1 JSR FLTFLX
00148 CB2E 2C .BYTE #2C
00149 CB2F A0 B1 DEFCLR LDY #01
00150 CB31 8C 19 C1 STY COLBYT
00151 CB34 4C 18 C1 JMP COLORR
00152 CB37 ;
00153 CB37 ; *****
00154 CB37 ;
00155 CB37 ; PLOT OR UNPLOT POINTS
00156 CB37 ;
00157 CB37 A9 00 PLOT2 LDA #0
00158 CB39 2C .BYTE #2C
00159 CB3A A9 B0 UNPLT2 LDA #B0
00160 CB3C 8D 3E 03 STA PLTFLG
00161 CB3F 20 CB CA JSR GETNUM
00162 CB42 8C 3F 03 STY XPLT
00163 CB45 8D 40 03 STA XPLT+1
00164 CB48 20 CB CA JSR GETNUM
00165 CB4B 8C 41 03 STY YPLT
00166 CB4E 4C 4B C1 JMP PLOTR
00167 CB51 ;
00168 CB51 ; *****
00169 CB51 ;
00170 CB51 ; DRAW LINES BETWEEN (X1,Y1) AND (X2,Y2)
00171 CB51 ; OR SHADED FACETS BETWEEN THREE POINTS
00172 CB51 ; (X1,Y1), (X2,Y2) AND (X3,Y3)
00173 CB51 ;
00174 CB51 A9 00 LINE2 LDA #0
00175 CB53 2C .BYTE #2C
00176 CB54 A9 B0 FACET2 LDA #B0
00177 CB56 8D 93 03 STA LINFAC
00178 CB59 20 CB CA JSR GETNUM
00179 CB5C 8C 4A 03 STY XMIN
00180 CB5F 8D 4B 03 STA XMIN+1
00181 CB62 20 CB CA JSR GETNUM
00182 CB65 8C 4C 03 STY YMIN
00183 CB68 20 CB CA JSR GETNUM
00184 CB6B 8C 4D 03 STY XMD
00185 CB6E 8D 4E 03 STA XMD+1
00186 CB71 20 CB CA JSR GETNUM
00187 CB74 8C 4F 03 STY YMD
00188 CB77 2C 93 03 BIT LINFAC
00189 CB7A 10 18 BPL LDRAW
00190 CB7C 20 CB CA JSR GETNUM
00191 CB7F 8C 50 03 STY XMAX
00192 CB82 8D 51 03 STA XMAX+1
00193 CB85 20 CB CA JSR GETNUM
00194 CB88 8C 52 03 STY YMAX
00195 CB8B 20 CB CA JSR GETNUM
00196 CB8E 8C 4A 03 STY VALUE
00197 CB91 4C E1 C4 JMP FACETR
00198 CB94 4C DB C2 LDRAW JMP LINER
00199 CB97 ;
00200 CB97 ; *****
00201 CB97 ;
00202 CB97 ; DRAW A SPHERE CENTERED AT (XCENT,YCENT)
00203 CB97 ; DEFAULT RADIUS IS LAST VALUE USED
00204 CB97 ;
00205 CB97 20 FB CA SPHER2 JSR CENTER
00206 CB9A 20 D2 CA JSR PCHCK
00207 CB9D 24 FB BIT DEFLAG
00208 CB9F 30 09 BMI SKIP1
00209 CBA1 20 9E AD JSR EVAEXP
00210 CBA4 20 AA B1 JSR FLTFLX
00211 CBA7 8C 77 03 STY RADIUS
00212 CBAA 4C C7 C7 SKIP1 JMP SPHERR
00213 CBAD ;
00214 CBAD ; *****
00215 CBAD ;
00216 CBAD ; DRAW A TOP-VIEW TOROID AT (XCENT,YCENT)
00217 CBAD ; DEFAULT INNER AND OUTER RADIUS ARE LAST USED
00218 CBAD ;
00219 CBAD 20 FB CA TORUS2 JSR CENTER
00220 CB80 20 E4 CA JSR GETTWO
00221 CB83 4C 0F C9 JMP TORUSR
00222 CB86 ;
00223 CB86 ; *****
00224 CB86 ;
00225 CB86 ; DRAW CYLINDERS WITH AXES HORIZONTAL OR
00226 CB86 ; VERTICAL. DEFAULT RADIUS AND "HALF-LENGTH"
00227 CB86 ; ARE LAST VALUES USED.
00228 CB86 ;
00229 CB86 A9 B0 VCYL2 LDA #B0
00230 CB88 2C .BYTE #2C
00231 CB89 A9 00 HCYL2 LDA #0
00232 CB8B 8D 83 03 STA HVFLAG
00233 CB8E 20 FB CA JSR CENTER
00234 CB91 20 D2 CA JSR PCHCK
00235 CB94 24 FB BIT DEFLAG
00236 CB96 30 0F BMI SKIP2
00237 CB98 20 9E AD JSR EVAEXP
```

(Continued on page 78)

Does your **ISAM**
run on **IBM,**
APPLE, DEC
and **AT&T**
computers?
c-tree does, and
you only **BUY**
IT ONCE!



c-tree™
BY FAIRCOM

2606 Johnson Drive
Columbia MO 65203

The company that introduced micros to B+ Trees in 1979 and created ACCESS MANAGER™ for Digital Research, now redefines the market for high performance, B+ Tree based file handlers. With c-tree™ you get:

- complete C source code written to K&R standards of portability
- high level, multi-key ISAM routines and low level B+ Tree functions
- routines that work with single-user and network systems
- no royalties on application programs

\$395 COMPLETE

Specify format:
5 1/4" PC-DOS 3 1/2" Mac
8" CP/M® 8" RT-11

for VISA, MC or COD orders, call
1-314-445-6833

Access Manager and CP/M are trademarks of Digital Research, Inc. Apple is a trademark of Apple Computer, Inc. c-tree and the circular disc logo are trademarks of FairCom IBM is a trademark of International Business Machines Corporation DEC is a trademark of Digital Equipment Corporation © 1984 FairCom

Circle no. 37 on reader service card.

\$5.00 C Compiler

Due to popular demand, **Dr. Dobb's Journal** has reprinted its most-asked-for C compiler articles by Ron Cain and J. E. Hendrix, each for only \$5.00.

Ron Cain's C compiler from sold-out 1980 issues #45 and #48 includes "A Small C Compiler for the 8080s" and "Runtime Library for the Small C Compiler."

The J. E. Hendrix reprint includes part two of "Small-C Compiler v.2" from sold out issue #75 and completes the first part of the compiler article from issue #74 which is included in Dr. Dobb's Bound Volume 7.

To Order: Enclose \$5.00 for each copy with this coupon and send to: Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303 Outside U.S., add \$2.00 per copy for shipping and handling.

Please send _____ copy(ies) of the Ron Cain Reprint, and
_____ copy(ies) of the J. E. Hendrix reprint to:

Name _____

Address _____

City _____ State _____ Zip _____

ALL REPRINT ORDERS MUST BE PREPAID.

Please allow 6-9 weeks for delivery.

103

C Programmers: Program three times faster with *Instant-C™*

Instant-C™ is an optimizing interpreter for the C language that can make programming in C three or more times faster than using old-fashioned compilers and loaders. The interpreter environment makes C as easy to use as Basic. Yet *Instant-C™* is 20 to 50 times faster than interpreted Basic. This new interactive development environment gives you:

Instant Editing. The full-screen editor is built into *Instant-C™* for immediate use. You don't wait for a separate editor program to start up.

Instant Error Correction. You can check syntax in the editor. Each error message is displayed on the screen with the cursor set to the trouble spot, ready for your correction. Errors are reported clearly, by the editor, and only one at a time.

Instant Execution. *Instant-C™* uses no assembler or loader. You can execute your program as soon as you finish editing.

Instant Testing. You can immediately execute any C statement or function, set variables, or evaluate expressions. Your results are displayed automatically.

Instant Symbolic Debugging. Watch execution by single statement stepping. Debugging features are built-in; you don't need to recompile or reload using special options.

Instant Loading. Directly generates .EXE or .CMD files at your request to create stand-alone versions of your programs.

Instant Floating Point. Uses 8087* co-processor if present.

Instant Compatibility. Follows K & R standards. Comprehensive standard library provided, with source code.

Instant Satisfaction. Guaranteed, or your money back. *Instant-C™* is available now, and works under PC-DOS, MS-DOS*, and CP/M-86*.

Find out how *Instant-C™* is changing the way that programming is done. *Instant-C™* is \$495. Call or write for more information.

Rational
Systems, Inc.

(617) 653-6194

P.O. Box 480

Natick, Mass. 01760

Trademarks: MS-DOS (Microsoft Corp.), 8087 (Intel Corp.), CP/M-86 (Digital Research, Inc.), Instant-C (Rational Systems, Inc.)

Circle no. 79 on reader service card.

Listing Five

```
00230 C8CB 20 AA B1 JSR FLTFLX
00239 C8CE 8C 77 03 STY RADIUS
00240 C8D1 20 C8 CA JSR GETNUM
00241 C8D4 8C 89 03 STY HLEN
00242 C8D7 4C 64 C8 SKIP2 JMP CYLNDL
00243 C8DA ;
00244 C8DA ; *****
00245 C8DA ;
00246 C8DA ; DRAW EDGE-VIEW TOROIDS WITH AXES HORIZONTAL
00247 C8DA ; OR VERTICAL
00248 C8DA ; INNER AND OUTER RADII ARE OPTIONAL
00249 C8DA ;
00250 C8DA A9 80 VTOR2 LDA #80
00251 C8DC 2C .BYTE #2C
00252 C8DD A9 00 HTOR2 LDA #0
00253 C8DF 8D 03 03 STA HVFLAG
00254 C8E2 20 F8 CA JSR CENTER
00255 C8E5 20 E4 CA JSR GETTWO
00256 C8E8 4C 8F C8 JMP EDGTOR
00257 C8EB ;
00258 C8EB ; *****
00259 C8EB ;
00260 C8EB ; DRAW INSIDE-VIEW TOROIDS, "SPOOLS",
00261 C8EB ; WITH AXES HORIZONTAL OR VERTICAL
00262 C8EB ; INNER AND OUTER RADII ARE OPTIONAL
00263 C8EB ;
00264 C8EB A9 80 VSFL2 LDA #80
00265 C8ED 2C .BYTE #2C
00266 C8EE A9 00 HSFL2 LDA #0
00267 C8F0 8D 03 03 STA HVFLAG
00268 C8F3 20 F8 CA JSR CENTER
00269 C8F6 20 E4 CA JSR GETTWO
00270 C8F9 4C 3B CA JMP SPOOLR
00271 C8FC .END
```

ERRORS = 00000

SYMBOL TABLE

SYMBOL VALUE

CENTER	CAF8	CH.COM	AEDF	CLEAR2	C80B	CLEARR	C12C
CLBRYT	C135	COLBYT	C119	COLOR2	C821	COLORR	C118
CYLNDL	C864	DEFCLR	C817	DEFCOL	C82F	DEFLAG	00F8
DFACUT	CAF8	EDGTOR	C80F	EVAEXP	AD9F	FACD2	0054
DFACETR	C4E1	FLTFLX	B1AA	GETNUM	CACB	GETTWO	C4E4
GRFOFF	C103	GRFON	C0E2	HCYL2	C8B9	HLEN	C8B9
HSFL2	C8EE	HTOR2	C8DD	HVFLAG	0383	LDRW	C14B
LINE2	C851	LNER	C20B	LINFAC	0393	NOMORE	CADF
ORIGIN	AC68	PCHECK	CAD2	PLOT2	C837	PLOTR	C14B
PLTFLG	033E	RADIUS	0377	RAM	0393	RI	026F
RD	039E	SKIF1	C6AA	SKIF2	C807	SPHER2	C897
SPHERR	C7C7	SPOOLR	C43B	TORUS2	C8AD	TORUSR	C90F
UNFLT2	C83A	VALUE	0344	VCYL2	C8B6	VSFL2	C8EB
VTOR2	C8DA	XCENT	036A	XMAX	0350	XMIN	034D
XMIN	034A	XPLT	033F	YCENT	036F	YMAX	0352
YMIN	034F	YMIN	034C	YPLT	0341		

END OF ASSEMBLY

End Listing Five

Listing Six

```
10 REM SHAPES DEMO
20
30 REM RICHARD L. RYLANDER 11/23/84 (REVISED 1/20/85 TO ADD LABELING)
40
50 GR=49378 :REM GRAPHICS MODE
60 TX=49411 :REM TEXT MODE
70
80 LB=893 :REM LEFT BOUND
90 RB=894 :REM RIGHT BOUND
100 UB=895 :REM UP BOUND
110 DB=896 :REM DOWN BOUND
120
130 REM FLAGS FOR VARIOUS DRAWING MODES
140
150 SH=830 :REM SHADE STYLE - 0=RANDOM, 1=HALFTONE
160 SC=839 :REM SCALING - 0=NORMAL (1:1), 1=SCALED (3:4) FOR SCREEN DISPLAY
170 LT=898 :REM LIGHTING - 0=NORMAL SINGLE-SOURCE, 1=BACKLIT ILLUMINATION
180
190 BO=53280 :REM BORDER COLOR
200
210 REM FUNCTION LOCATIONS
220
230 CL=51979 :REM CLEAR BITMAP AREA
240 CO=52001 :REM FILL COLOR MAP
250
260 SP=52119 :REM SPHERE
270 TR=52141 :REM TOP-VIEW TOROID
280 VC=52150 :REM CYLINDER (AXIS VERTICAL)
290 HC=52153 :REM CYLINDER (AXIS HORIZONTAL)
300 VT=52186 :REM EDGE-VIEW TOROID (AXIS VERTICAL)
310 HT=52189 :REM EDGE-VIEW TOROID (AXIS HORIZONTAL)
320 VS=52203 :REM INSIDE-VIEW TOROID ("SPOOL") (AXIS VERTICAL)
330 HS=52206 :REM INSIDE-VIEW TOROID ("SPOOL") (AXIS HORIZONTAL)
340
350 REM DRAW SAMPLES OF PRIMITIVE SHAPES WITH LABEL BENEATH EACH
360
370 POKE SH,1 :REM HALFTONE SHADING
380 POKE SC,1 :REM USE SCREEN SCALING (FOR UNDISTORTED SPHERES, ETC.)
390 POKE LT,0 :REM USE NORMAL (NO BACKLIT) ILLUMINATION
400 SYS(CO),16*11 :REM CLEAR SCREEN (DEFAULT - FILL BITMAP WITH 0'S)
410 SYS(CO),16*11 :REM COLOR COMBINATION - DARK GRAY (1) DOTS ON WHITE (1)
420 REM ON MOST COLOR MONITORS
430 POKE BO,1 :REM WHITE BORDER TO MATCH BACKGROUND
440 SYS(GR)
450 REM "H-CYLINDER" (CYLINDER)
460 SYS(VC),200,199 :REM NO SIZE PARAMETERS GIVEN, DEFAULT (PREVIOUS) ARE USED
470 CH=21 :REM "H-CYLINDER" (CYLINDER)
480 SYS(TR),200,199,15,38
490 CH=32 :REM "H-TOROID" (TOROID)
500 SYS(VT),40,64
510 RW=23 :REM "H-TOROID" (TOROID)
520 SYS(HT),120,64
530 CH=11 :REM "H-TOROID" (TOROID)
540 SYS(HS),200,64,5,100
550 CH=22 :REM "H-SPOOL" (SPOOL)
560 SYS(VS),200,64
570 CH=32 :REM "H-SPOOL" (SPOOL)
580 SYS(VS),200,64
```

```
530 POKE 198,0:WAIT 198,1:POKE 198,0
540 REM WAIT FOR A KEY TO BE PRESSED
550 :
560 REM DRAW TWO "GOBLET" ONE WITH HALFTONE, THE OTHER RANDOM SHADING.
570 :
580 SYS(CO),SYS(CO),16*11+1
590 RW=14:CM=15:AS="COMPARISON":GOSUB 1900:RW=15:CM=19:AS="OF":GOSUB 1900
600 RW=16:CM=14:AS="SHADE STYLES":GOSUB 1900:RW=18:CM=14:AS="HALFTONE"
610 GOSUB 1900:RW=20:CM=15:AS="RANDOM" :>:GOSUB 1900
620 POKE LB,255:POKE RB,255:POKE UB,49:POKE DB,255 :REM CLIP AT SCREEN TOP
630 SYS(SP),80,190,80
640 POKE UB,51:POKE DB,51 :REM CLIP AT JUNCTION WITH SPHERE FOR SMOOTH SEAM
650 SYS(VS),80,69,10,130
660 POKE DB,9:POKE UB,8
670 SYS(VT),80,9,25,45
680 POKE SH,0 :REM SWITCH TO RANDOM SHADING
690 POKE LB,255:POKE RB,255:POKE UB,49:POKE DB,255
700 SYS(SP),240,190,80
710 POKE UB,51:POKE DB,51
720 SYS(VS),240,69,10,130
730 POKE DB,9:POKE UB,8
740 SYS(VT),240,9,25,45
750 POKE 198,0:WAIT 198,1:POKE 198,0
760 :
770 REM DRAW "WINE" SCENE
780 POKE LT,1 :REM BACKLIT ILLUMINATION
790 POKE SH,1 :REM HALFTONE SHADING FOR MOST (RANDOM "LABEL" ON BOTTLE)
800 SYS(CO),SYS(CO),255 :REM FILL BITMAP WITH 1'S ("SET" BACKGROUND)
810 POKE BO,0 :REM BLACK BORDER TO MATCH BACKGROUND
820 RW=0:CM=0:MD=2:AS="DRAWING WITH":GOSUB 1900:RW=1:CM=2:AS="BACKLIT"
830 GOSUB 1900:RW=2:CM=0:AS="AGAINST A SET":GOSUB 1900
840 RW=3:CM=2:AS="BACKGROUND":GOSUB 1900
850 RW=1:CM=26:AS="COLORS ADDED":GOSUB 1900
860 RW=2:CM=25:AS="TO SELECT AREAS":GOSUB 1900
870 REM DRAW BOTTLE
880 POKE UB,0:POKE DB,255:POKE LB,255:POKE RB,255
890 SYS(VT),150,10,30,50
900 POKE UB,255:SYS(VC),150,70,50,60
910 POKE DB,8:SYS(VT),150,130,6,20
920 POKE DB,55:POKE UB,0:SYS(VS),150,204,15,181
930 POKE UB,255:SYS(VC),150,221,16,17
940 :
950 REM DRAW WINE GLASS
960 POKE UB,20:SYS(SP),80,120,60
970 POKE UB,35:POKE DB,34:SYS(VS),80,34,10,110
980 :
990 REM DRAW SOME GRAPES
1000 SYS(SP),8,8,8 :REM OTHER "GRAPES" WILL BE SAME RADIUS - JUST GIVE POSITIONS
1010 SYS(SP),20,8:SYS(SP),40,8:SYS(SP),12,20:SYS(SP),30,20:SYS(SP),25,16
1020 :
1030 REM DRAW AN APPLE BY A PAIR OF EDGE-VIEW TOROIDS AND A SPHERE SECTION
1040 SYS(VT),260,29,0,50:SYS(VT),260,77
1050 POKE UB,43:POKE DB,43:SYS(SP),260,54,60
1060 REM PUT STEM ON APPLE
1070 POKE RB,0:POKE DB,0:SYS(TR),272,104,10,15
1080 REM ADD A LEAF BY A SPHERE SECTION
1090 POKE DB,255:POKE RB,0:SYS(SP),256,119,15
1100 REM ADD A RANDOM SHADED "LABEL" TO THE BOTTLE
1110 POKE DB,255:POKE RB,255:POKE LB,6
1120 POKE SH,0:SYS(VC),150,72,50,48
1130 :
1140 REM ADD COLORS TO "WINE" SCENE
1150 SYS(CO),12:REM REPLACE WHITE "HOLES" (BACKGROUND) WITH MEDIUM GRAY
1160 X1=0:Y1=200:X2=100:Y2=239:DC=0:BC=0:GOSUB 1700:REM HIDE TEXT IN CORNERS
1170 X1=180:Y1=200:X2=319:Y2=239:DC=0:BC=0:GOSUB 1700
1180 X1=200:Y1=110:X2=319:Y2=180:DC=0:BC=1:GOSUB 1700
1190 X1=240:Y1=110:X2=255:Y2=150:BC=5:GOSUB 1700
1200 X1=260:Y1=110:X2=270:Y2=135:BC=9:GOSUB 1700
1210 X1=1:Y1=1:X2=40:Y2=30:BC=4:GOSUB 1700
1220 X1=140:Y1=205:X2=180:Y2=235:BC=7:GOSUB 1700
1230 X1=145:Y1=25:Y2=195:Y2=115:BC=6:GOSUB 1700
1240 POKE 198,0:WAIT 198,1:POKE 198,0
1250 :
1260 REM "COFFEE AND DONUTS"
1270 POKE SH,0 :REM RANDOM SHADING ON DONUTS
1280 SYS(CO),16*11+1:SYS(CO),POKE BO,1:REM WHITE BORDER TO MATCH BACKGROUND
1290 POKE LB,255:POKE RB,255:POKE UB,255:POKE DB,255 :REM NO INITIAL CLIPPING
1300 SYS(VT),60,20,20,60
1310 POKE RB,29:SYS(VT),99,60:POKE RB,255
1320 SYS(TR),188,180
1330 REM ADD COFFEE CUP (HALFTONE SHADING)
1340 POKE SH,1:POKE UB,0:SYS(VT),188,20:POKE UB,255
1350 POKE DB,0:SYS(TR),278,110,20,40
1360 POKE DB,255:POKE UB,0:POKE LB,0
1370 SYS(TR),240,90,50,78:POKE LB,255:SYS(SP),248,110,10:POKE UB,255
1380 SYS(VC),300,100,10,10
1390 SYS(VC),188,77,60,57
1400 POKE DB,0:SYS(VT),188,134,40,60
1410 REM ADD COLOR - FIRST MAKE SCREEN BROWN DOTS ON WHITE THEN MAKE CUP GREEN
1420 SYS(CO),1+16*9 :REM 1=WHITE BACKGROUND, 9=BROWN DOTS
1430 X1=130:Y1=1:X2=319:Y2=136:BC=1:DC=5:GOSUB 1700
1440 X1=250:Y1=144:X2=319:Y2=144:GOSUB 1700
1450 POKE 198,0:WAIT 198,1:POKE 198,0
1460 :
1470 REM DRAW "LINKED" TOROIDS BY REDRAWING OVERLAP AREAS USING CLIPPING
1480 POKE LT,0 :REM BLUE DOTS ON WHITE, NO BACKLIT
1490 SYS(CO),SYS(CO),1+16*6 :REM 1=WHITE BACKGROUND, 6=BLUE DOTS
1500 POKE SH,0 :REM RANDOM SHADING
1510 SYS(TR),244,84,48,70
1520 SYS(TR),160,84:SYS(TR),76,84
1530 SYS(CO),118,156:SYS(TR),202,156
1540 REM ADD PROPER OVERLAP SECTIONS
1550 POKE RB,0:POKE DB,0:SYS(TR),160,84:POKE RB,255:POKE LB,0
1560 SYS(TR),76,84:POKE DB,255:POKE UB,0:SYS(TR),118,156
1570 POKE LB,255:POKE RB,0:SYS(TR),202,156:POKE LB,27
1580 POKE DB,0:POKE UB,255:SYS(TR),160,84
1590 POKE DB,0:POKE UB,255:SYS(TR),160,84
1600 POKE LB,255:POKE UB,27:SYS(TR),244,84
1610 POKE LB,0:POKE RB,27:POKE UB,255:SYS(TR),244,84
1620 POKE 198,0
1630 GET AS:IF AS="" THEN 1590
1640 SYS(TX):POKE BO,14:REM RETURN TO TEXT MODE
1650 END
1660 :
1670 REM SUBROUTINE FOR ADDING COLOR TO DIFFERENT SCREEN AREAS
1680 REM REMEMBER THAT COLOR BOUNDARIES MUST CORRESPOND TO CHARACTER BOUNDARIES
1690 REM DEFINE A RECTANGULAR AREA BY LOWER-LEFT AND UPPER-RIGHT COORDINATES
1700 REM (X1,Y1)=LOWER-LEFT POINT, (X2,Y2)=UPPER-RIGHT POINT
1710 REM THE CORNER POINTS CAN BE ARBITRARY BUT COLOR CHANGE WILL BE IN THE
1720 REM SMALLEST CHARACTER-CELL BOUNDED RECTANGLE THAT INCLUDES THOSE POINTS
1730 REM Y-COORDINATES MUST BE "UNSCALED" IF SCALE FLAG IS SET:
1740 IF PEEK(SC) THEN Y1=(Y1-1)*213/256+Y2*(Y2-1)*213/256
1750 REM COLOR TO BE POKE IN IS GIVEN BY VARIABLE CC="COMPOSITE COLOR"
1760 REM WHERE CC=16*DC + BC [DC=DOT COLOR, BC=BACKGROUND COLOR NUMBERS]
1770 CC=16*DC+BC
1780 FOR IY=INT(Y1/8) TO INT(Y2/8)
1790 FOR IX=INT(X1/8) TO INT(X2/8)
1800 FOR IY=INT(Y1/8) TO INT(Y2/8)
```

(Continued on page 80)

THE fifth generation language

P R O L O G

Implementing the full Edinburgh Syntax as described by Clocksin and Mellish. Recognized by Japan as providing unparalleled opportunity for artificial intelligence.

Applications:

- ▲ The highest level of a heirarchical robotic control system.
- ▲ Machine recognition of natural language.
- ▲ Expert systems and knowledge engineering.

Optional:

- ▲ Type VM L Prolog makes possible the execution of AI applications previously only possible on a mainframe.
- ▲ Invisible compilation to a semantic network provides the flexibility of the interpreted mode and the speed of a compiler.
- ▲ Virtual memory and a sophisticated cache algorithm limited only by the DOS, typically 2 gigabytes.
- ▲ Syntax superset with many extensions such as pattern specified insertion and deletion, clause indexing, robotic control.
- ▲ Unlimited number of resident and virtual modules, each up to 2 gigabytes in size.

**special
Educational
Package
\$29.95**
(MSDOS)

▲ THE most cost efficient package in the industry.
\$30-\$500 (MSDOS version) Also available for Xenix, Unix, CP/M68K

▲ VISA, Mastercard, AMEX

▲ call or write for brochure



automata design assoc.

1570 Arran Way Dresher, PA 19025 technical: (215)646-4894 orders: (215)355-5400

Circle no. 130 on reader service card.

PROLOG V

INTRODUCTORY OFFER

Valid through
May 31, 1985

**ONLY
\$69⁹⁵**

Examine the
documentation for
30 days and return
with disk still sealed
for full refund, if not
fully satisfied.

Prolog Artificial Intelligence Programming Language

Full Prolog as defined in Clocksin & Mellish, *Programming in Prolog*, Springer-Verlag, Berlin Heidelberg New York, 1981.

Complete documentation with primer and working-program examples.
Interpreter for IBM PCs® and compatibles.

IBM PC® is a registered trademark of IBM Corporation.

PHONE ORDERS:
(619) 483-8513



**CHALCEDONY
SOFTWARE**

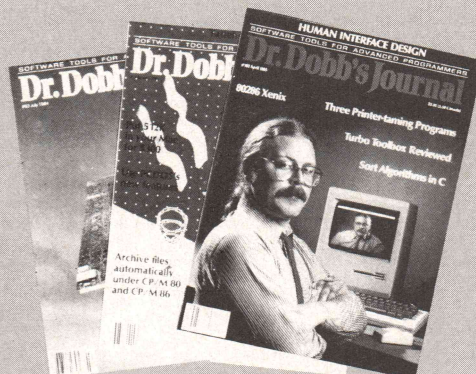
5580 LA JOLLA BLVD.
SUITE 126 F.
LA JOLLA, CA
92037

<input type="checkbox"/>	PAYMENT ENCLOSED \$ _____
<input type="checkbox"/>	CHARGE MY: <input type="checkbox"/> MasterCard <input type="checkbox"/> Visa
Card No. _____	Exp. Date _____
Signature _____	
Mr./Mrs./Ms. _____ (please print full name)	
Address _____	
City/State/Zip _____	842DD

Circle no. 21 on reader service card.

AVAILABLE DDJ BACK ISSUES

1982	1983	1984	1985
No. 68—June	No. 76—Feb.	No. 87—Jan.	No. 99—Jan.
No. 69—July	No. 77—March	No. 88—Feb.	No. 100—Feb.
No. 70—Aug.	No. 78—April	No. 90—April	No. 101—March
No. 71—Sept.	No. 80—June	No. 91—May	No. 102—April
No. 72—Oct.	No. 81—July	No. 92—June	
No. 73—Nov.	No. 82—Aug.	No. 94—Aug.	
	No. 83—Sept.	No. 95—Sept.	
	No. 84—Oct.	No. 96—Oct.	
	No. 85—Nov.	No. 97—Nov.	
	No. 86—Dec.	No. 98—Dec.	



TO ORDER: send \$3.50 per issue to: Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303.

Name _____

Address _____

City _____

State _____

Zip _____

103

Drawing on the C-64 (Listing Continued, text begins on page 50)

Listing Six

```

1760 POKE 34752:IX=40:IV,CC
1770 NEXT:RETURN
1780
1790 REM SUBROUTINE TO ADD TEXT TO GRAPHIC SCREEN.
1800 REM "RW" AND "CM" ARE THE ROW (0-24) AND COLUMN (0-39) COORDINATES FOR THE
1810 REM FIRST LETTER OF THE TEXT STRING TO BE PRINTED.
1820 REM THE TEXT STRING ITSELF IS ASSIGNED TO "A$".
1830 REM "MD" IS THE MODE FOR THE PRINTING. FIVE MODES ARE ALLOWED:
1840 REM 1 - NORMAL ("BLACK" LETTERS ON "WHITE" BACKGROUND)
1850 REM 2 - REVERSED ("WHITE" LETTERS ON "BLACK" BACKGROUND)
1860 REM 3 - SET ("BLACK") LETTERS "OR'ED" WITH BACKGROUND
1870 REM 4 - UNSET ("WHITE") LETTERS "AND'ED" WITH BACKGROUND
1880 REM 5 - SET LETTERS "XOR'ED" WITH BACKGROUND
1890 :
1900 SB=40952:TB=54272:IF (MD AND 1) THEN TB=53248:REM SCREEN AND TEXT BASE ADDR'S
1910 OS=320:RW=0:CM=REM OFFSET FROM CHARACTER SCREEN BASE
1920 POKE 56334,PEEK(56334)AND 254:REM DISABLE 1RD TIMER
1930 POKE 1,PEEK(1)AND 251:REM SWITCH CHARACTER ROM IN
1940 L=LEN(A$):FOR N=1 TO L:NB=N*OS+SB
1950 X=ASC(MID$(A$,N,1)):IF X>63 THEN X=X-64
1960 TC=TB+8+X
1970 ON MD GOTO 1980,1980,1990,2000,2010
1980 POKE 53231,36:GOTO 2020
1990 POKE 53231,17:GOTO 2020
2000 POKE 53231,49:GOTO 2020
2010 POKE 53231,81
2020 POKE 252,NB/256:POKE 251,NB-256*INT(NB/256)
2030 POKE 254,TC/256:POKE 253,TC-256*INT(TC/256)
2040 SYS(53221):NEXT
2050 POKE 1,PEEK(1)OR 4:POKE 56334,PEEK(56334)OR 1:REM RESTORE TO NORMAL
2060 RETURN

```

READY.

End Listing Six

Listing Seven

```

10 REM "STELLATION" DRAW A SMALL STELLATED DODECAHEDRON IN VARIOUS ORIENTATIONS
20 REM AND STYLES RICHARD L. RYLANDER 12/5/84
30 :
40 GR=49378 :REM GRAPHICS MODE
50 TX=49411 :REM TEXT MODE
60 BO=53260 :REM BORDER COLOR
70 :
80 REM INITIALIZE A FEW STYLE PARAMETERS
90 POKE 839,1 :REM SCALE (3:4) FOR UNDISTORTED SCREEN DISPLAY
100 POKE 871,0 :REM FACET EDGE/LINE MODE (0=DRAW, 1=ERASE)
110 SH=0:0 :REM SHADE STYLE (0=RANDOM, 1=HALFTONE)
120 EG=868 :REM EDGES FLAG (0=NORMAL, 1=ADD LINES TO FACET EDGES)
130 :
140 REM FUNCTION LOCATIONS
150 CL=51977 :REM CLEAR BITMAP
160 CO=52001 :REM FILL COLOR MAP
170 FC=52052 :REM DRAW A SHADED FACET
180 KS=53081 :REM DO KEYED SORT
190 :
200 XC=160:YC=120 :REM (SCALED) SCREEN CENTER COORDINATES
210 :
220 PRINT"(SC) *****"
230 PRINT" * SMALL STELLATED DODECAHEDRON *****"
240 PRINT" *****"
250 PRINT"(CD) (CD) SELECT SHADING STYLE:"
260 PRINT" R=RANDOM, H=HALFTONE"
270 INPUT"(CD) YOUR CHOICE H(C)L(C)L(C)L(C)":"A$
280 POKE SH,0:IF A$="H" THEN POKE SH,1
290 PRINT"(CD) (CD) SELECT STYLE FOR DRAWING:"
300 PRINT"(CD) N - NORMAL:PRINT(CD) E - EDGES EMPHASIZED"
310 PRINT"(CD) W - WIRE FRAME (HIDDEN LINES REMOVED)"
320 INPUT"(CD) YOUR CHOICE N(C)L(C)L(C)L(C)":"A$
330 POKE EG,0:W=0:IF A$="N" THEN 360
340 POKE EG,1:IF A$="W" THEN W=1
350 :
360 PRINT"(CD) READING VERTEX DATA"
370 VN=32:DIM F%(VN-1,2):REM VN = NUMBER OF VERTICES
380 FOR N=0 TO VN-1:READ P%(N,0),P%(N,1),P%(N,2):NEXT
390 :
400 PRINT"(CD) ENTER X, Y, AND Z ANGLES FOR ROTATION"
410 PRINT" (ANGLES IN DEGREES)"
420 INPUT X,Y,Z
430 J=3.14159265/180:X=X*J:Y=Y*J:Z=Z*J
440 X=COS(Y)*COS(Z)-SIN(X)*SIN(Y)*SIN(Z):X1=COS(Y)*SIN(Z)+SIN(X)*SIN(Y)*COS(Z)
450 X2=-COS(X)*SIN(Y)+Y0=-COS(X)*SIN(Z):Y1=COS(X)*COS(A):Y2=SIN(X)
460 Z0=SIN(Y)*COS(Z)+SIN(X)*COS(Y)*SIN(Z)
470 Z1=SIN(Y)*SIN(Z)-SIN(X)*COS(Y)*COS(Z):Z2=COS(X)*COS(Y)
480 PRINT"(CD) PERFORMING ROTATION"
490 FOR N=0 TO VN-1
500 X=P%(N,0):Y=P%(N,1):Z=P%(N,2)
510 P%(N,0)=X0=X+X1*Y+Y2*Z:P%(N,1)=Y0=Y+Y1*Y+Y2*Z:P%(N,2)=Z0=Z+Z1*Y+Z2*Z:NEXT
520 :
530 FA=60:REM TOTAL NUMBER OF FACETS
540 DIM F%(FA/2,2),SH(F%/2),Z%(FA/2),K%(FA/2)
550 PRINT" READING CONNECTION DATA"
560 VF=-1:REM VF = # VISIBLE FACETS
570 FOR N=1 TO FA
580 VF=VF+1
590 FOR I=0 TO 2:READ F%(VF,1):NEXT
600 REM CALCULATE COMPONENTS OF NORMAL VECTOR TO DETERMINE VISIBILITY
610 Z=(P%(F%(VF,2),0)-P%(F%(VF,1),0))*P%(F%(VF,0),1)-P%(F%(VF,1),1))
620 Z2=(P%(F%(VF,0),0)-P%(F%(VF,1),0))*P%(F%(VF,2),1)-P%(F%(VF,1),1))
630 IF Z2<0 THEN 720:REM FACET NOT VISIBLE, GO ON
640 X=(P%(F%(VF,2),1)-P%(F%(VF,1),1))*P%(F%(VF,0),2)-P%(F%(VF,1),2))
650 X=X-(P%(F%(VF,2),1)-P%(F%(VF,1),1))*P%(F%(VF,2),2)-P%(F%(VF,1),2))
660 Y=(P%(F%(VF,2),2)-P%(F%(VF,1),2))*P%(F%(VF,0),0)-P%(F%(VF,1),0))
670 Y=Y-(P%(F%(VF,2),2)-P%(F%(VF,1),2))*P%(F%(VF,2),0)-P%(F%(VF,1),0))
680 NC=SOR(X*X+Y*Y+Z*Z):REM LENGTH OF NORMAL VECTOR
690 SH(VF)=26*(2*Z+X*Y)/NC
700 SH(VF)=(SH(VF)+64)*(SH(VF)+64)/256:REM RAISED COSINE SHADING
710 GOTO 730
720 VF=VF-1
730 NEXT
740 :
750 PRINT" SCALING TO DISPLAY SIZE"
760 Y=0:FOR N=0 TO VN-1:IF ABS(F%(N,1))>Y THEN Y=ABS(F%(N,1))
770 NEXT:IS=1/9*Y
780 FOR N=0 TO VN-1:P%(N,1)=S*F%(N,1)+YC:P%(N,0)=S*P%(N,0)+XC:NEXT
790 :
800 REM FIND AVERAGE Z FOR EACH FACET
810 FOR N=0 TO VF
820 Z%(N)=(P%(F%(N,0),2)+P%(F%(N,1),2)+P%(F%(N,2),2))/3:NEXT
830 :
840 PRINT" SORTING FACETS ACCORDING TO AVG 'Z' "
850 POKE 140,VF
860 K%(0)=K%(0):POKE 251,PEEK(71):POKE 252,PEEK(72)
870 Z%(0)=Z%(0):POKE 253,PEEK(71):POKE 254,PEEK(72)
880 SYS(KS)
890 :

```

```

900 REM DRAW FACETS (ACCORDING TO Z DEPTH SINCE NOT CONVEX)
910 SYS(GR):SYS(CO):SYS(CL):POKE BO,1
920 FOR N=0 TO VF:FA=K%(N)
930 IF W1 THEN SH(FA)=64
940 X0=P%(F%(FA,0),0):Y0=P%(F%(FA,0),1):X1=P%(F%(FA,1),0):Y1=P%(F%(FA,1),1)
950 X2=P%(F%(FA,2),0):Y2=P%(F%(FA,2),1)
960 SYS(FC):X0,Y0,X1,Y1,X2,Y2,SH(FA)
970 NEXT
980 POKE 198,0
990 GET A:IF A$="" THEN 990
1000 SYS(TX):POKE BO,14:END
1010 :
1020 REM VERTEX DATA
1030 DATA 1000,618,0, 1000,-618,0, -1000,618,0, -1000,-618,0
1040 DATA 0,1000,618, 0,1000,-618, 0,-1000,618, 0,-1000,-618
1050 DATA 618,0,1000, -618,0,1000, 618,0,-1000, -618,0,-1000
1060 DATA 618,0,236, 618,0,-236, -618,0,236, -618,0,-236
1070 DATA 236,618,0, -236,618,0, 236,618,0, -236,618,0
1080 DATA 0,236,618, 0,-236,618, 0,236,618, 0,-236,618
1090 DATA 382,382,382, 382,382,-382, 382,-382,382, -382,-382,382
1100 DATA -382,382,382, -382,382,-382, -382,-382,382, -382,-382,-382
1110 :
1120 REM CONNECTION DATA
1130 DATA 0,12,13, 0,13,25, 0,25,16, 0,16,24, 0,24,12
1140 DATA 1,12,26, 1,26,18, 1,18,27, 1,27,13, 1,13,12
1150 DATA 2,15,14, 2,14,26, 2,26,17, 2,17,29, 2,29,15
1160 DATA 3,14,15, 3,15,31, 3,31,19, 3,19,30, 3,30,14
1170 DATA 4,16,17, 4,17,20, 4,20,20, 4,20,24, 4,24,16
1180 DATA 5,17,16, 5,16,25, 5,25,22, 5,22,29, 5,29,17
1190 DATA 6,19,18, 6,18,26, 6,26,21, 6,21,30, 6,30,19
1200 DATA 7,19,19, 7,19,31, 7,31,23, 7,23,27, 7,27,19
1210 DATA 8,20,21, 8,21,26, 8,26,12, 8,12,24, 8,24,20
1220 DATA 9,21,20, 9,20,28, 9,20,14, 9,14,30, 9,30,21
1230 DATA 10,23,22, 10,22,25, 10,25,12, 10,12,27, 10,27,23
1240 DATA 11,22,23, 11,23,31, 11,31,15, 11,15,29, 11,29,22

```

READY.

End Listing Seven

Listing Eight

```

00001 0000 : KEYSORT - RELOCATABLE BUBBLE SORT USING KEY ARRAY
00002 0000 : POINTING TO INTEGER ARRAY
00003 0000 :
00004 0000 : RICHARD L. RYLANDER 1/12/85
00005 0000 :
00006 0000 : ORIGIN=4CF59 : 53081. (FOLLOWING DOS 5.1)
00007 0000 :
00008 0000 : KB = #FB : 251. POINTER TO KEY ARRAY
00009 0000 : ZB = #FD : 253. POINTER TO DATA ARRAY
00010 0000 : MAX = #FC : 140. POKE WITH MAX ARRAY INDEX
00011 0000 : #AC
00012 0000 : TOPDIS = #AD
00013 0000 : FLAG = #AE
00014 0000 : NXTFLG = #61
00015 0000 : CRNT = #62
00016 0000 : REPEAT = #64
00017 0000 :
00018 0000 : **=ORIGIN
00019 CF59 :
00020 CF59 A0 FF : INIT LDY #FF : INITIALIZE KEY ARRAY
00021 CF5B C8 : INLOOP INY
00022 CF5C 78 : TAY
00023 CF5D 91 FB : STA (KB),Y
00024 CF5F C5 8C : CMP MAX
00025 CF61 D0 FB : BNE INLOOP
00026 CF63 :
00027 CF63 85 AD : SORT STA TOPDIS : 'A' HOLDS 'MAX'
00028 CF65 A5 AD : LOOP1 LDA TOPDIS
00029 CF67 85 AC : LDA TOP
00030 CF69 A2 00 : LDY #0
00031 CF6B 86 61 : STX NXTFLG
00032 CF6D 86 AE : STX FLAG
00033 CF6F 86 64 : LOOP2 STX REPEAT
00034 CF71 :
00035 CF71 :
00036 CF71 : GET BOTH BYTES OF INTEGER POINTNED TO BY
00037 CF71 : 'KEY' ELEMENT. RETURN WITH MSB ON STACK
00038 CF71 : AND LSB IN THE ACCUMULATOR
00039 CF71 BA : GETINT TXA
00040 CF72 AB : TAY
00041 CF73 B1 FB : LDA (KB),Y
00042 CF75 BA : ASL A
00043 CF76 98 04 : BCC LOAD
00044 CF78 C6 61 : DEC NXTFLG
00045 CF7A E6 FE : INC ZB+1
00046 CF7C AB : LOAD TAY
00047 CF7D B1 FD : LDA (ZB),Y
00048 CF7F 4B : PHA
00049 CF80 CB FD : INY
00050 CF81 B1 FD : LDA (ZB),Y
00051 CF83 24 61 : BIT NXTFLG
00052 CF85 10 04 : BPL NODEC
00053 CF87 E6 61 : INC NXTFLG
00054 CF89 C6 FE : DEC ZB+1
00055 CF8B E4 64 : NODEC CPX REPEAT
00056 CF8D D0 00 : BNE ORDER
00057 CF8F 05 62 : PLA
00058 CF91 68 : STA CRNT
00059 CF92 85 63 : STA CRNT+1
00060 CF94 E8 : INX
00061 CF95 D0 DA : BNE GETINT
00062 CF97 :
00063 CF97 : COMPARE INTEGERS OBTAINED THROUGH KEY ARRAY
00064 CF97 : IF 'CURRENT' >= 'NEXT' THEN SWAP KEY
00065 CF97 : ELEMENTS, ELSE CONTINUE
00066 CF97 :
00067 CF97 :
00068 CF97 : ORDER CMP CRNT
00069 CF99 68 : PLA
00070 CF9A E5 63 : SBC CRNT+1
00071 CF9C 50 02 : BVC TEST
00072 CF9E 49 00 : EOR #0
00073 CFA0 10 13 : TEST BPL NOSWAP
00074 CFA2 8A : SWAP TXA
00075 CFA3 AB : TAY
00076 CFA4 86 AD : STX TOPDIS
00077 CFA6 B1 FB : LDA (KB),Y
00078 CFA8 4B : PHA
00079 CFA9 8B : DEY
00080 CFAA B1 FB : LDA (KB),Y
00081 CFAC C9 00 : INY
00082 CFAD 91 FB : STA (KB),Y
00083 CFAF 68 : PLA
00084 CFB0 8B : DEY
00085 CFB1 91 FB : STA (KB),Y

```

(Continued on page 82)

Dr. Dobb's Journal

A. Your primary job function: (Check one only)

- ☐ Company management (Pres., V.P., Treas., Owner, Gen. Mgr., Mktg. Dir.)
- ☐ Computer systems management (V.P. EDP, MIS Director, Data Processing Mgr., Data Communications Mgr., Network Planner)
- ☐ Engineering management (V.P. Engrg., Chief Engr., Tech. Director, Dir. R&D)
- ☐ Systems integrators (Systems Designer, Project Engr., Systems Application Engr., Technical Staff Members)
- ☐ Consultants (Computer/EDP/Data Communications Consultant)
- ☐ Educators (Educational users and instructors of computer technology)
- ☐ Systems/Programming specialists—mini-micro systems
- ☐ Other (Please specify) _____

B. Which languages are you MOST interested in?

- ☐ BASIC ☐ C ☐ PL/I
- ☐ Fortran ☐ LISP ☐ APL
- ☐ COBOL ☐ Prolog ☐ Logo
- ☐ Pascal ☐ Ada ☐ Smalltalk
- ☐ Modula-2 ☐ Forth ☐ Other _____

C. What is the operating system?

- ☐ CP/M (or derived)
- ☐ UNIX (or derived)
- ☐ MS-DOS (or derived)
- ☐ Other _____

D. Please indicate which of the following microcomputers you currently own and/or plan to buy in the next 12 months.

	Own	Plan to Buy
Apple	<input type="checkbox"/>	<input type="checkbox"/>
Commodore	<input type="checkbox"/>	<input type="checkbox"/>
Digital Equipment/DEC	<input type="checkbox"/>	<input type="checkbox"/>
Heath/Zenith	<input type="checkbox"/>	<input type="checkbox"/>
Hewlett-Packard	<input type="checkbox"/>	<input type="checkbox"/>
IBM	<input type="checkbox"/>	<input type="checkbox"/>
Macintosh	<input type="checkbox"/>	<input type="checkbox"/>
Radio Shack/Tandy TRS 80	<input type="checkbox"/>	<input type="checkbox"/>
Texas Instruments	<input type="checkbox"/>	<input type="checkbox"/>
Other (Specify)	<input type="checkbox"/>	<input type="checkbox"/>
None	<input type="checkbox"/>	<input type="checkbox"/>

E. What best describes the work you do with this microcomputer?

- ☐ Business functions
- ☐ Software/Hardware development
- ☐ Scientific/Engineering/R&D applications

F. Are you the decision maker or very influential in computer-related purchases at work?

- ☐ Yes ☐ No

G. Is your company a dealer, distributor, or systems house for microcomputers?

- ☐ Yes ☐ No

H. Subscriber

- ☐ Yes ☐ No

To obtain information about products or services mentioned in this issue, circle the appropriate number listed below. Use bottom row to vote for best article in issue. One card per person.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81
82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108
109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135

Articles: 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214

This offer expires August 31, 1985

Name _____ Phone (____) _____

Address _____

City/State/Zip _____

Dr. Dobb's Journal

A. Your primary job function: (Check one only)

- ☐ Company management (Pres., V.P., Treas., Owner, Gen. Mgr., Mktg. Dir.)
- ☐ Computer systems management (V.P. EDP, MIS Director, Data Processing Mgr., Data Communications Mgr., Network Planner)
- ☐ Engineering management (V.P. Engrg., Chief Engr., Tech. Director, Dir. R&D)
- ☐ Systems integrators (Systems Designer, Project Engr., Systems Application Engr., Technical Staff Members)
- ☐ Consultants (Computer/EDP/Data Communications Consultant)
- ☐ Educators (Educational users and instructors of computer technology)
- ☐ Systems/Programming specialists—mini-micro systems
- ☐ Other (Please specify) _____

B. Which languages are you MOST interested in?

- ☐ BASIC ☐ C ☐ PL/I
- ☐ Fortran ☐ LISP ☐ APL
- ☐ COBOL ☐ Prolog ☐ Logo
- ☐ Pascal ☐ Ada ☐ Smalltalk
- ☐ Modula-2 ☐ Forth ☐ Other _____

C. What is the operating system?

- ☐ CP/M (or derived)
- ☐ UNIX (or derived)
- ☐ MS-DOS (or derived)
- ☐ Other _____

D. Please indicate which of the following microcomputers you currently own and/or plan to buy in the next 12 months.

	Own	Plan to Buy
Apple	<input type="checkbox"/>	<input type="checkbox"/>
Commodore	<input type="checkbox"/>	<input type="checkbox"/>
Digital Equipment/DEC	<input type="checkbox"/>	<input type="checkbox"/>
Heath/Zenith	<input type="checkbox"/>	<input type="checkbox"/>
Hewlett-Packard	<input type="checkbox"/>	<input type="checkbox"/>
IBM	<input type="checkbox"/>	<input type="checkbox"/>
Macintosh	<input type="checkbox"/>	<input type="checkbox"/>
Radio Shack/Tandy TRS 80	<input type="checkbox"/>	<input type="checkbox"/>
Texas Instruments	<input type="checkbox"/>	<input type="checkbox"/>
Other (Specify)	<input type="checkbox"/>	<input type="checkbox"/>
None	<input type="checkbox"/>	<input type="checkbox"/>

E. What best describes the work you do with this microcomputer?

- ☐ Business functions
- ☐ Software/Hardware development
- ☐ Scientific/Engineering/R&D applications

F. Are you the decision maker or very influential in computer-related purchases at work?

- ☐ Yes ☐ No

G. Is your company a dealer, distributor, or systems house for microcomputers?

- ☐ Yes ☐ No

H. Subscriber

- ☐ Yes ☐ No

To obtain information about products or services mentioned in this issue, circle the appropriate number listed below. Use bottom row to vote for best article in issue. One card per person.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81
82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108
109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135

Articles: 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214

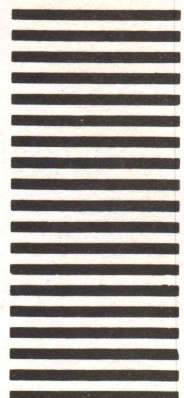
This offer expires August 31, 1985

Name _____ Phone (____) _____

Address _____

City/State/Zip _____

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



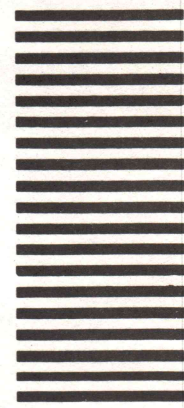
BUSINESS REPLY MAIL
FIRST CLASS PERMIT #27346, PHILADELPHIA, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS
Dr. Dobb's Journal
P.O. BOX 13851
PHILADELPHIA, PA 19101



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL
FIRST CLASS PERMIT #27346, PHILADELPHIA, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS
Dr. Dobb's Journal
P.O. BOX 13851
PHILADELPHIA, PA 19101

TOTAL CONTROL:

FORTH: FOR Z-80®, 8086, 68000, and IBM® PC

Complies with the New 83-Standard

**GRAPHICS • GAMES • COMMUNICATIONS • ROBOTICS
DATA ACQUISITION • PROCESS CONTROL**

● **FORTH** programs are instantly portable across the four most popular microprocessors.

● **FORTH** is interactive and conversational, but 20 times faster than BASIC.

● **FORTH** programs are highly structured, modular, easy to maintain.

● **FORTH** affords direct control over all interrupts, memory locations, and i/o ports.

● **FORTH** allows full access to DOS files and functions.

● **FORTH** application programs can be compiled into turnkey COM files and distributed with no license fee.

● **FORTH** Cross Compilers are available for ROM'ed or disk based applications on most microprocessors.

Trademarks: IBM, International Business Machines Corp.; CP/M, Digital Research Inc.; PC/Forth+ and PC/GEN, Laboratory Microsystems, Inc.

FORTH Application Development Systems include interpreter/compiler with virtual memory management and multi-tasking, assembler, full screen editor, decompiler, utilities and 200 page manual. Standard random access files used for screen storage, extensions provided for access to all operating system functions.

Z-80 FORTH for CP/M® 2.2 or MP/M II, \$100.00;

8080 FORTH for CP/M 2.2 or MP/M II, \$100.00;

8086 FORTH for CP/M-86 or MS-DOS, \$100.00;

PC/FORTH for PC-DOS, CP/M-86, or CCPM, \$100.00; **68000 FORTH** for CP/M-68K, \$250.00.

FORTH + Systems are 32 bit implementations that allow creation of programs as large as 1 megabyte. The entire memory address space of the 68000 or 8086/88 is supported directly.

PC FORTH + \$250.00

8086 FORTH + for CP/M-86 or MS-DOS \$250.00

68000 FORTH + for CP/M-68K \$400.00

Extension Packages available include: software floating point, cross compilers, INTEL 8087 support, AMD 9511 support, advanced color graphics, custom character sets, symbolic debugger, telecommunications, cross reference utility, B-tree file manager. Write for brochure.



Laboratory Microsystems Incorporated

Post Office Box 10430, Marina del Rey, CA 90295

Phone credit card orders to (213) 306-7412



Circle no. 55 on reader service card.

YOU NEED A GOOD LIBRARY



COMPLETE SOURCES NO ROYALTIES

COMPREHENSIVE C Power Packs include over 1000 functions which provide an integrated environment for developing your applications efficiently. "This is a beautifully documented, incredibly comprehensive set of C Function Libraries."

— Dr. Dobb's Journal, July 1984

USEFUL "...can be used as an excellent learning tool for beginning C Programmers..."

— PC User's Group of Colorado, Jan. 1985

FLEXIBLE Most Compilers and all Memory Models supported.

RECOMMENDED "I have no hesitation in recommending it to any programmer interested in producing more applications code, using more of the PC capabilities, in much less time."

— Microsystems, Oct. 1984

■ **PACK 1: Building Blocks I** \$149
DOS, Keyboard, File, Printer, Video, Async

■ **PACK 2: Database** \$399
B-Tree, Virtual Memory, Lists, Variable Records

■ **PACK 3: Communications** \$149
Smartmodem™, Xon/Xoff, X-Modem, Modem-7

■ **PACK 4: Building Blocks II** \$149
Dates, Textwindows, Menus, Data Compression, Graphics

■ **PACK 5: Mathematics I** \$99
Log, Trig, Random, Std Deviation

■ **PACK 6: Utilities I** \$99
(EXE files)
Arc, Diff, Replace, Scan, Wipe

*Master Card/Visa, \$7 Shipping, Mass. Sales Tax 5%

ASK FOR FREE DEMO DISKETTE

**NOVUM
ORGANUM
INC.**

**SOFTWARE
HORIZONS
INC.**

165 Bedford St., Burlington, MA 01803
(617) 273-4711

Circle no. 90 on reader service card.

An Optimizing, hassle-free C Compiler for the 8086-8088.

Ecosoft's Eco-C, the performance leader among full C compilers for the Z80, is now available for the 8086-8088 running under MSDOS (2.0 or later). Eco-C has features not found in any other 8086-8088 C compiler.

★ Over 100 library functions. Since they follow UNIX standards, your programs are highly portable in "both" directions ("up" to a UNIX machine or "down" to our Z80 compiler). This means new markets for your software at minimum development cost.

★ A single library. No more "dual" libraries and trying to remember what has to be linked with what. Your programs automatically take advantage of an 8087 if one is present at runtime.

★ A single floating point answer. No more "fuzzy floating point": your programs produce the same answers whether the floating point is done in hardware (8087) or software.

★ Error messages in English. No more cryptic numbers to look up. We tell you where the error occurred, what was found there, and what should have been there.

★ Strict syntax parsing. LINT is going to uncover fewer surprises because our parser looks hard at the details.

★ Efficient code. The optimizer pass of the compiler generates assembler code in Intel mnemonics that rivals that produced by compilers costing twice as much.

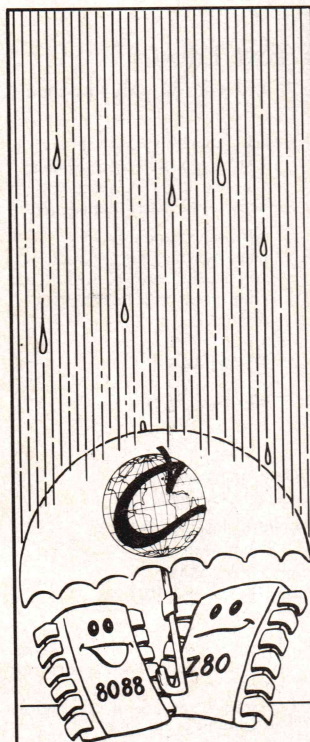
The price of the Eco-C compiler is \$250.00 (all versions), including the user's manual, and is designed for use with Microsoft's MASM (or compatible) assembler and linker. When ordering, please specify disk format and whether you want the Z80-CP/M or 8088-MSDOS version of Eco-C.

Ecosoft Inc.
6413 N. College Avenue
Indianapolis, IN 46220
(317) 255-6476



Eco-C (Ecosoft), MSDOS (Microsoft), UNIX (Bell Labs), CP/M (Digital Research), Z80 (Zilog), 8086, 8087, 8088 (Intel).

Circle no. 35 on reader service card.



Drawing on the C-64

Listing Eight

(Listing Continued, text begins on page 50)

```

00005 CF83 E6 AE      INC FLAG
00006 CF85 E4 AC      NOSWAP CPX TOP
00007 CF87 D8 B6      BNE LOOP2
00008 CF89 A5 AE      LDA FLAG
00009 CF8B D0 AB      BNE LOOP1
00010 CF8D
00011 CF8D
00012 CF8D
00013 CF8D
00014 CF8D A6 BC      UNFACK LDX MAX
00015 CF8F E8         INX
00016 CF90 CA         PKLOOP DEX
00017 CF91 8A         TXA
00018 CF92 AB         TAY
00019 CF93 51 FB      LDA (KB),Y
00020 CF95 4B         PHA
00021 CF96 8A         TXA
00022 CF97 0A         ASL A
00023 CF98 09 01      ORA #1
00024 CF9A 90 04      BCC STORE
00025 CF9C E6 61      INC STORE
00026 CF9E E6 FC      INC KB+1
00027 CF9F AB         STORE TAY
00030 CF01 68         PLA
00031 CF02 91 FB      STA (KB),Y
00032 CF04 A9 00      LDA #0
00033 CF06 B8         DEY
00034 CF07 91 FB      STA (KB),Y
00035 CF09 A5 61      LDA NXTFLG
00036 CF0B F0 04      BEQ OK
00037 CF0D C6 61      DEC NXTFLG
00038 CF0F C6 FC      DEC KB+1
00039 CF11 8A         OK TXA
00040 CF12 D0 DC      BNE PKLOOP
00041 CF14 60         DONE RTS
00042 CF15

```

ERRORS = 00000

SYMBOL TABLE

SYMBOL	VALUE	SYMBOL	VALUE	SYMBOL	VALUE	SYMBOL	VALUE
CRNT	0062	DONE	CFE4	FLAG	00AE	GETINT	CF71
INIT	CF59	INLOOP	CF5B	KB	00FB	LOAD	CF7C
LOOP1	CF65	LOOP2	CF6F	MAX	00BC	NODEC	CF8B
NOSWAP	CF85	NXTFLG	0061	OK	CFE1	ORDER	CF97
ORIGIN	CF59	PKLOOP	CF90	REPEAT	0064	SHORT	CF63
STORE	CFD0	SWAP	CFA2	TEST	CFA0	TOP	00AC
TOPDIS	00AD	UNFACK	CF8D	ZB	00FD		

END OF ASSEMBLY

End Listing Eight

Listing Nine

```

00001 0000      ; "WRITE" RICHARD L. RYLANDER
00002 0000      ; 12/30/84
00003 0000      ; REVISED 1/19/85 - ORIGIN MOVED TO #CFE5 (53221.)
00004 0000
00005 0000      ; PUT TEXT CHARACTERS ON GRAPHIC SCREEN
00006 0000      ; (UNDER BASIC ROM) IN VARIOUS STYLES
00007 0000
00008 0000      ;
00009 CF85 A5 01  WRITE LDA #01      ; PUT CODE AFTER DOS 5.1
00010 CF87 29 FE  AND #FE        ; SWITCH OUT BASIC ROM
00011 CF89 85 01  STA #01
00012 CF8B A0 07  LDY #7
00013 CF8D B1 FD  LOOP LDA (#FD),Y  ; READ CHARACTER BYTE
00014 CF8F 31 FB  AND (#FB),Y    ; MODIFY W/SCREEN BYTE
00015 CFF1 91 FB  STA (#FB),Y    ; STORE IN SCREEN
00016 CFF3
00017 CFF3      ;
00018 CFF3      ; POKE NEW LOGICAL OPERATOR TO REPLACE
00019 CFF3      ; 'AND' (53231.) FOR DIFFERENT STYLES
00020 CFF3      ; ORA=17, BIT (NOP)=36, AND=49, EOR=81.
00021 CFF3 8B      ;
00022 CFF4 10 F7  DEY
00023 CFF6 A5 01  BPL LOOP
00024 CFF8 07 01  LDA #01      ; RESTORE BASIC ROM
00025 CFFA 85 01  ORA #1
00026 CFFC 60      STA #01
00027 CFFD      RTS

```

ERRORS = 00000

SYMBOL TABLE

SYMBOL	VALUE	SYMBOL	VALUE
LOOP	CF8D	WRITE	CF85

END OF ASSEMBLY

End Listings

Fortran Scientific Subroutine Package

Contains Approx. 100 Fortran Subroutines Covering:

1. Matrix Storage and Operations
2. Correlation and Regression
3. Design Analysis
4. Discriminant Analysis
5. Factor Analysis
6. Eigen Analysis
7. Time Series
8. Nonparametric Statistics
9. Distribution Functions
10. Linear Analysis
11. Polynomial Solutions
12. Data Screening

Sources Included

\$295.00

FORLIB-PLUS™

Contains three assembly coded LIBRARIES plus support, FORTRAN coded subroutines and DEMO programs.

The three LIBRARIES contain support for GRAPHICS, COMMUNICATION, and FILE HANDLING/DISK SUPPORT. An additional feature within the graphics library is the capability of one fortran program calling another and passing data to it. Within the communication library, there are routines which will permit interrupt driven, buffered data to be received. With this capability, 9600 BAUD communication is possible. The file handling library contains all the required software to be DOS 3.0 PATHNAME compatible.

STRINGS & THINGS™

Support for CHARACTER MANIPULATION (string support), SHELL, BATCH, MUSIC, CMD LINE, and ENVIRON CTRL.

\$69.95 each

P.O. Box 2517
Cypress, CA 90630



(714) 894-6808

California residents, please add 6% sales tax.

Versions available for IBM Professional Fortran or MICROSOFT 3.2 Fortran

Circle no. 1 on reader service card.

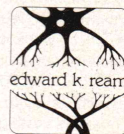
C Source Code

RED

Full Screen Text Editor

IBM PC, Kaypro, CP/M 80 and CP/M 68K systems.

- RED is fast! RED uses all of your terminal's special functions for best screen response. RED handles files as large as your disk automatically and quickly.
 - RED is easy to use for writers or programmers. RED's commands are in plain English.
 - RED comes with complete source code in standard C. RED has been ported to mainframes, minis and micros.
 - RED comes with a Reference Card and a Reference Manual that provides everything you need to use RED immediately.
 - RED is unconditionally guaranteed. If for any reason you are not satisfied with RED your money will be refunded promptly.
- RED: \$95**
Manual: \$10



Call or write today for more information:

Edward K. Ream
1850 Summit Avenue
Madison, WI 53705
(608) 231-2952

To order:

Either the BDS C compiler or the Aztec CII compiler is required for CP/M80 systems. Digital Research C compiler v1.1 is required for CP/M 68K systems. No compiler is required for IBM or Kaypro systems.

Specify both the machine desired (IBM, Kaypro or CP/M) and the disk format described (8 inch CP/M single density or exact type of 5 1/4 inch disk).

Send a check or money order for \$95 (\$105 U.S. for foreign orders). Sorry, I do NOT accept phone, credit card, or COD orders. Please do not send purchase orders unless a check is included. Your order will be mailed to you within one week.

Dealer inquiries invited.

JOIN US AT NCC '85

Today's Access Code to Tomorrow's Technology.

**Featuring Keynote Speaker Admiral B.R. Inman.
NCC '85, July 15-18, Chicago, Illinois.**

The "Leading Edge" conference ...

Don't miss the 1985 National Computer Conference, July 15-18 at McCormick Place and McCormick Place West, Chicago, Illinois, a key industry forum for new product and services introductions.

NCC '85 features a Keynote Address by Admiral B.R. Inman, President and CEO of Microelectronics and Computer Technology Corporation of Austin, Texas.

His topic, "Managing the Creation and Commercialization of Technology" will deal with the industry-wide challenge of converting technical advances into commercial successes.

A challenge everyone in the information technology field faces.

More than just a trade show ...

At NCC '85 you get more than just exhibits. You can attend informative Technical Sessions on subjects such as artificial intelligence and software systems. You can also attend Professional Development Seminars, and special events like Pioneer Day and our

Early Bird Reception. Plus, you can explore, experience and exchange ideas while gathering important information for future buying decisions.

Top names in the industry will be there—like IBM, Cullinet and McGraw Hill—showcasing their latest innovations and hottest new developments.

Whatever your special interest or area of expertise, you'll find the state-of-the-art in hardware, software and services at NCC '85!

Tomorrow's technology is just a phone call away ...

To attend NCC '85 all you have to do is pick up the phone and call toll-free 800-NCC-1985. (Or fill out the coupon below.) And, if you register early, you get a \$25 discount on the Full Conference Registration fee!

Don't miss this opportunity to be on the "Leading Edge." Plan on attending NCC '85!

**CALL TOLL-FREE 800-NCC-1985
(Or fill out the coupon below.)**



1985 National Computer Conference

Technology's Expanding Horizons

☐ **Yes!** I want to attend NCC '85. Please send me a registration form today!

Name _____

Title _____

Company _____

Address _____

City _____

State _____

Zip _____

Mail to: NCC '85 • 1899 Preston White Dr. • Reston, VA 22091 TB

OR CALL TOLL-FREE 800-NCC-1985 AND REGISTER TODAY! Visa, MasterCard, and American Express accepted. Register early and save \$25!

A Compiler Written in Prolog

by G. A. Edgar

Why would anyone write a compiler in a language like Prolog? Actually, the compiler seems to have come out rather well. I wrote it primarily as a learning exercise for Micro-PROLOG.

Micro-PROLOG

I first heard of the Prolog programming language in connection with the Japanese effort to develop the "fifth generation" in computing. At the time, I read a few things about Prolog, such as the book by Clocksin and Mellish.¹ My impression was that Prolog is an interesting language (I was right), but that it probably is not practical to implement PROLOG on a microcomputer (was I ever wrong!).

substantial effort for a programmer experienced only in Pascal, C, or PL/I. The tutorial² is published separately as a book; if you are curious, it should be available in most large university libraries.

After I began to get a feel for the language, I used it to solve a few logic puzzles. For example, Lewis Carroll's logic book³ includes some puzzles at the end as a teaser for a later book that was never published. They are much too complicated to solve simply by common sense reasoning, but I can now say definitively that "All of the Monitors are awake."

If Prolog were good only for logic problems, it would not be of wide interest. So I looked around for some other relatively easy task to use as a

The early returns on our March special issue on Prolog are encouraging—so much so that we're offering this excursion into programming in logic.

D. E. Cortesi, in "Dr. Dobb's Clinic" for May 1984, gave a half-page description of Micro-PROLOG. I wrote for information on it and by July was running it on my CP/M Z80 computer. Micro-PROLOG is also available for MSDOS, PCDOS, CP/M-86, and Unix.

Micro-PROLOG comes with an IBM PC-size loose-leaf manual with more than 200 pages. This is a complete reference to the language but not a good tool for learning. Micro-PROLOG, however, also comes with a 400-page soft-cover tutorial, which is a good place for a beginner to start. Learning the language will take a

programming exercise. A compiler for VALGOL fit the need nicely. It improved my understanding of Micro-PROLOG and my understanding of what a compiler does.

This compiler is described below. It translates VALGOL I, also described below, into 8080 assembly language. Of course, you would learn more by undertaking such a programming project yourself than you will by examining mine, but it serves as proof that you can use Micro-PROLOG for something unexpected.

Using Micro-PROLOG

Before I describe the compiler, let me give a brief description of how Micro-PROLOG differs from other versions of Prolog.

G. A. Edgar, 107 W. Dodridge St.,
Columbus, OH 43202

Upper- and lower-case letters are considered different, and the hyphen is treated like a letter. Variables are X, Y, Z, x, y, z, or one of these six letters followed by a positive integer, such as X132 or z22. Other strings of letters and numbers are identifiers for constants.

Micro-PROLOG consists primarily of a pattern matcher, plus a few other built-in routines, such as a routine for I/O. The way Micro-PROLOG looks depends on which of the front ends you are using. Each front-end program is written in Prolog. If you are using the SIMPLE front end, one-place predicates can be given either in prefix form

alphanumeric(X)

or in postfix form

X alphanumeric

Two-place predicates can be given either in prefix form

LESS(@ X)

or in infix form

@ LESS X

Predicates with three or more arguments must be in prefix form. Thus, a Prolog program entered via SIMPLE tends to look like this:

```

labelused(0)
X alphanumeric if X alphabetic
X alphanumeric if X digit
X alphabetic if @ LESS X and X
LESS [
X alphabetic if ' LESS X and X
LESS {

```

One useful object in Micro-PROLOG is the list. The notation for a list is to enclose the items between parentheses: (A B C 1 2 3). Some of the items might be lists themselves: (A (B C) (1 2 3)).

The 16-bit versions of Micro-PROLOG include a front-end supervisor that imitates the DEC-10 Prolog:

```

labelused(0).
alphanumeric(X) :- alphabetic(X).

```

alphanumeric(X) :- digit(X).

The form in which an operator is used is declared using the types fx, xf, xfy, and so on (see Clocksin and Mellish, page 91). DEC-10 lists are built using square brackets: [a,b,c].

The standard Micro-PROLOG syntax simply uses lists. In this regard, it is somewhat like LISP. The list (labelused 0), which indicates the predicate name "labelused" applied to the argument "0," is an example of an atom. A clause is a list of atoms.

You add things to the data base in this way:

```

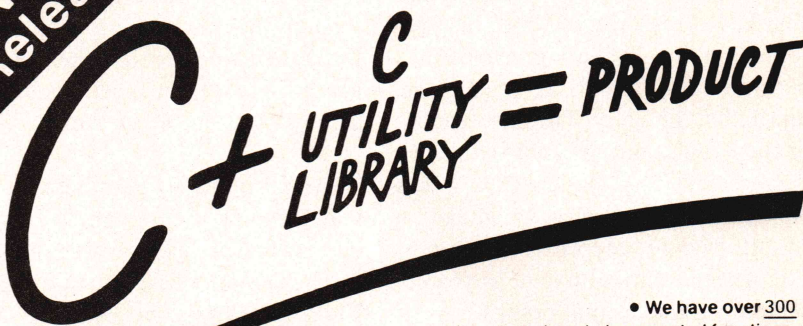
((labelused 0))
((alphanumeric X) (alphabetic X))
((alphanumeric X) (digit X))

```

This may seem confusing at first, but after a while it becomes more or less routine.

This minimal introduction to the Micro-PROLOG interpreter should be enough for you to gain some understanding of the VALGOL compil-

New Release



• We have over 300
complete, tested, and, documented functions.
All source code and demo programs are included.

• The library was specifically designed for software
development on the IBM PC, XT, AT and compatibles. There are no royalties.

• Over 95% of the source code is written in C. Experienced programmers
can easily "customize" functions. Novices can learn from the thorough comments.

We already have the functions you are about to write

Concentrate on software development—not writing functions.

THE C UTILITY LIBRARY includes:


- Best Screen Handling Available • Windows • Full Set of Color Graphics Functions • Better String Handling Than Basic • DOS Directory and File Management • Execute Programs, DOS Commands and Batch Files • Complete Keyboard Control • Extensive Time Date Processing • Polled ASYNC Communications • General DOS BIOS gate • Data Entry • And More •

• The Library is compatible with: Lattice, Microsoft, Computer Innovations, Mark Williams and DeSmet. Available Soon: Digital Research, Aztec and Wizard.

C Compilers: Lattice C—\$349, Computer Innovations C86—\$329; Mark Williams C—\$449.

C UTILITY LIBRARY \$185. Special prices on library & compiler packages.

Order direct or through your dealer. Specify compiler when ordering. Add \$4.00 shipping for UPS ground, \$7.00 for UPS 2-day service. NJ residents add 6% sales tax. Master Card, Visa, check or P.O.



ESSENTIAL SOFTWARE, INC

P.O. Box 1003 Maplewood, New Jersey 07040 914 762-6605

Circle no. 36 on reader service card.

er, but you will need a much more complete knowledge of Micro-PROLOG to understand all of the details.

VALGOL I

The language VALGOL I (very small ALGOL?) is a derivative of ALGOL-60. D. V. Schorre described it in 1964 as a sample language for the compiler-writing language META II. His paper⁴ was reprinted in *Dr. Dobb's Journal* (April 1980), which is where I found it. I had worked with VALGOL I in connection with the compiler-writing language Meta4 (in a SIG/M disk), so it seemed natural to use it for the same thing in Prolog.

VALGOL I has a few peculiarities. The keywords that usually are typeset in boldface here are preceded by a period (**.begin**, **.end**, **.if**, **.then**, **.else**, **.until**, **.do**, **.integer**). This also applies to the equals sign for comparing two expressions (**=**). VALGOL I has only one data type, namely **.integer**, a 16-bit two's-complement number. The assignment statement is the reverse from the normal order (**5 =: x** assigns the value 5 to x). Arithmetic allows addition, subtraction, and multiplication, but there is no unary minus sign, so if you want -2 you must write either 0 - 2 or 65534.

A more complete description of the syntax appears below in the description of the compiler.

Using the VALGOL I Compiler

The VALGOL I compiler is in a file called VALGOL.LOG. A typical use, to compile the short VALGOL program in the file P.VAL, is shown in Listing One (page 89). The command shown is in the standard Micro-PROLOG syntax. If the SIMPLE front end is also loaded, you could use the command:

is("P.VAL" compile "P.ASM")

The compiler prints out the input file P.VAL on the console as it works. The result of the compilation in the sample is P.ASM, included here as Listing Four (page 96).

If you are familiar with the Intel mnemonics for the 8080, you may find it instructive to compare the VALGOL input with the assembly language output to see how the compiler works.

The VALGOL Definition

Take a look at the compiler in Listing Two (page 90). I have repeated part of the compiler using the SIMPLE syntax in Listing Three (page 96). SIMPLE is unsuitable for the complete listing for several reasons. In a few places in the compiler, I have used "meta-variables," which are not supported by SIMPLE. A SIMPLE listing also would not keep the comments and the formatting, making it confusing to read. So try to understand instead the list notation used for the standard syntax of Micro-PROLOG in Listing Two.

The lists preceded by a ? are to be executed immediately when they are loaded; the others are to be stored in the data base. Because the predicate symbol (/ *) is a special one that is always true, a list in the file of the form

?((/ * anything goes here)) will serve as a comment. (This is a clumsy feature of Micro-PROLOG.)

The compiler listing is in two parts. The first part is independent of the particular syntax of VALGOL I and could be used equally well with many other programming languages. The second, below a horizontal line, is a VALGOL-specific part. In one sense, the part above the line implements an interpreter that runs the compiler described below the line. Let's examine a portion of the VALGOL-specific part of the code.

The message line, like all these lines, becomes a statement in the Prolog data base. The predicate symbol of the statement is "message," and the single argument is the string enclosed in quotation marks. The compiler will look for a message fact in the data base and print it out when a compilation starts.

The next line is the syntax line. This tells the compiler that the primary unit of syntax is the program.

The next statement is a description of a program. First comes the keyword **.begin**. When this is matched, seven opening lines are sent to the output file. (I have put the VALGOL syntax description on the left-hand side of the page and the ASM output description on the right-hand side. A VALGOL I compiler for another processor would

keep the same left-hand side and change the right-hand side.) Then an opt-declaration (optional declaration, see below) is followed by one or more statements separated by semicolons. The multiple predicate allows matching of its arguments zero or more times. Finally comes the keyword **.end**. When this is matched, the final JMP is inserted in the output file, followed by the four subroutines and the stack space.

The opt-declaration could be a declaration followed by a semicolon, or (if that cannot be found) it could be empty, which matches the input file automatically. Inclusion of the "empty" alternative is a way to allow for no declaration. A declaration is the keyword **.integer**, followed by one or more identifiers separated by commas. For each identifier, the output file reserves a two-byte storage location.

A statement is one of the following:

(1) An I/O statement, either

edit(expression , 'string')¹

which will send (expression) spaces and then the (string) to the console, or

print

which will send an end-of-line to the console.

(2) An assignment statement:

expression =: variable

(3) A loop:

.until expression
.do statement

A value of zero for the expression is considered false, and a nonzero value is true.

(4) A conditional statement:

.if expression
.then statement
.else statement

The **.else** is not optional.

(5) A block, which consists of the keyword **.begin**, followed by an optional declaration, followed by zero or more statements separated by semicolons, followed by the keyword

We're getting hardnosed at Softway.

From now on MATIS™ is only \$49⁹⁵!

(MATIS, the complete User Interface development tool has been selling for \$150.)

Why the radical price cut?

We decided after looking over the competition that MATIS had so many advantages it should be made available to more programmers. We decided to compete aggressively so you could easily afford to have MATIS in **your** bag of tricks. We hear from MATIS users in the USA and France that it is a truly loveable product. Sooo...we're running this big ad to promote our new low price.

MATIS windows are beautiful.

Display any portion of any screens you create at any point in your program. Scroll in any direction manually with cursor keys...or automatically.

And the screens are HUMUNGEOUS!

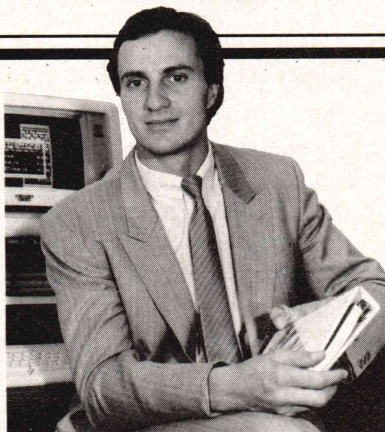
MATIS screens can be just about as big as you want...up to 65,534 rows by 65,534 columns! The number of screens is only limited by available memory.

Print big MATIS screens directly.

One command sends your screens to your printer with no need to program special routines when your virtual screen is bigger than your terminal screen.

User input fields are a snap.

Creating fields for data entry is easy with no limit to size or number by screen. Request for input separately or in blocks.



Denis Moran
President, Softway, Inc.

Control your keyboard with MATIS.

It keeps track of keys that are pressed during the execution of your program and lets you assign specific functions to selected keys.

Control that screen too!

MATIS is extremely versatile and flexible when it comes to controlling lines, columns, fields, and text. They can be modified, transferred, displayed or moved with a single command. All video attributes are supported: color, reverse video, blinking...you name it, you got it.

Want an interactive screen builder?

You've got it with MATIS. It's called "MATPAGE"™ and it lets you create and modify any of your screens in an interactive mode.

MATIS adds over 70 routines to your program.

Written in Assembler, MATIS routines are fast and powerful giving your program improved efficiency and enhanced visual appeal, while they reduce its size and maintenance worries. And MATIS separates screen design from the core of your program.

MATIS is unique.

We don't think there's a single program that combines as many tools in one package as completely or as well as MATIS. It interfaces with Interpreted and Compiled BASIC (Microsoft), C (Lattice, Microsoft, Aztec), PASCAL (IBM, Microsoft) and ASSEMBLER. All you need is an IBM* PC/XT or true compatible under DOS, 128k or RAM, monochrome or color monitor.

You get an easy to follow no-frills manual and a 30-Day Money Back Guarantee.

Late News: MATIS/T™ for TURBO-PASCAL** only \$29.95

An indispensable add-on at a dynamite price. What more can we say?

Denis Moran
Denis Moran

™ MATIS, MATIS/T, & MATPAGE are Trademarks of Softway, Inc.

*IBM is a Reg. Trademark of IBM Corp.

**Turbo Pascal is a Reg. Trademark of Borland International

Softway, Inc.

24-Hour Credit Card Orders By Phone:

1 (800) 227-2400 EXT 989 In California: 1(800) 772-2666 EXT 989

Please ship the following at once. I understand there is a 30-day money back guarantee.

_____ Copies of MATIS at \$49.95 plus \$3 shpg. \$ _____

_____ Copies of MATIS/T for TURBO PASCAL at \$29.95 plus \$2 shpg. \$ _____

California residents, add 6½% sales tax \$ _____

☐ I like to read specs, so send me a folder. Total \$ _____

Name _____

Address (please no P.O. Box) _____

City/State/ZIP _____

Phone (_____) _____ Signature _____

Make payment by money order, check, or charge card ☐ VISA ☐ MASTER CARD

Number _____ Exp. _____

Distributed in Europe By: MICRO APPLICATION SOFTWARE • 147 Avenue Paul Doumer, 92500 RUEIL MALMAISON FRANCE, Tel (1) 732.92.54

.end. The null statement

.begin .end

is allowed.

The following test for equality is allowed:

expression . = expression

This has value 1 if true and 0 if false. Expressions can be built up from variables, numbers, the operations +, -, *, and parentheses. Notice how the syntax near the end of Listing Two is arranged to take into account the precedence of these operations.

The Language-Independent Part

Now let's look at the first part of Listing Two (part of this is repeated in SIMPLE syntax as Listing Three). When Micro-PROLOG attempts to verify a compile atom, it looks first in the data base for a message fact; if it finds one, it verifies the assertion (P x), which is always true but which has the side effect of printing x on the console. The assertion PP is also always true and starts a new line at the console. The compile atom continues as it prints X -> Y, where X is the first argument (presumably the filename of the VALGOL input file) and Y is the second argument (the ASM output file).

Next the compiler deletes all facts about "labeledusd" (which may be left from a previous compile) and starts over with (labeledusd 0). It does something similar with "last-shown," which keeps track of how much of the file has been shown on the console.

Then the compiler opens the input file X and adds a fact to the data base so that later it can find the name of the input file. Similarly, it opens the output file Y. It skips white-space characters in the input file, starting at the beginning and ending at location Z1 (x in Listing Three). Then it finds the syntax fact to tell it what to start compiling with and feeds that to the Q routine, which carries out the actual compile.

Finally, the compiler closes the files and prints out a concluding message. If compilation fails for some reason, control goes to the second compile clause, and it prints out the error message.

Now look at Q. This short routine implements the recursive-descent compile, according to the code in the last half of the compiler. The first two arguments are the before and after file positions for both the input and output files. Whenever Q has to backtrack, the files automatically are rewound to the appropriate point for another try.

Notice the use of the variable Z (the third argument of Q) as the predicate symbol in a search of the data base. This is called a "meta-variable" in Micro-PROLOG: the name of the predicate to be verified is supplied as a variable, not as a constant. This powerful feature of Micro-PROLOG may not be available in other Prologs. The X1 after the vertical line, which indicates the rest of the list, is another meta-variable. The remainder of this half of the code comprises various subroutines for use in the compile.

This is a very brief description of only a small part of this compiler. A complete understanding requires a greater familiarity with the many features and quirks of the Micro-PROLOG package.

Conclusion

The compiler shown in Listing Two is

written in the standard syntax of Micro-PROLOG. As such, it should run under any version of Micro-PROLOG. I would be interested in hearing from readers who get it to work under other versions. (Of course, the output would still be in 8080 assembly language.)

This compiler runs rather slowly. The common belief is that interpreters are slow, and this is an interpreter within an interpreter. The 10-line sample compilation in Listing One takes about 30 sec on my 4 MHz Z80. Because I wrote the compiler primarily as a programming exercise, I have not been overly concerned with a remedy for the speed problem. Probably the bottleneck occurs in the input routines that process one character at a time. For a faster compiler, routines such as match, id, or number would be recoded in assembly language. The Micro-PROLOG manual has instructions on how to link such new built-in functions in assembly code with Micro-PROLOG.

References

1. W. F. Clocksin and C. S. Mellish, *Programming in Prolog*, Springer-Verlag, 1981.
2. K. L. Clark and F. G. McCabe, *Micro-PROLOG: Programming in Logic*, Prentice-Hall, 1984.
3. Lewis Carroll, *Symbolic Logic*, 4th edition, 1896 (Reprinted by Dover Publications, 1958).
4. D. V. Schorre, "META II, a syntax-oriented compiler writing language," *Proc. 19th Nat. Conf. ACM*, 1964 (Reprinted in *Dr. Dobb's Journal*, April 1980, pp. 17-21; listings pp. 22-25).

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 195.

Quelo™ 68000 Software Development Tools

68000/68010 Assembler Package

Assembler, linker, object librarian and extensive indexed typeset manuals.

Conforms to Motorola structured assembler, publication M68KASM[4]. Macros, cross reference and superb load map, 31 character symbols.

Optimized for CP/M-80, -86, -68K, MS-DOS, PC-DOS... \$ 595

Portable Source written in "C"..... \$1495

Complete 68000 Development Package for MS-DOS

Lattice 68000 "C" Compiler and Quelo 68000 Assembler Package..... \$1095

68200 Assembler Package

Assembler and linker for Mostek MK68200.

Optimized for CP/M-80, MS-DOS, PC-DOS..... \$ 595

For more information contact

Quelo Inc.

Patrick Adams
2464 33rd W. Suite #173
Seattle, WA 98199
Phone (206) 285-2528
telex II (TWX) 9103338171

COD, Visa, MasterCard
CP/M, TM DRI, MS-DOS TM Microsoft, PC-DOS TM IBM.

Circle no. 20 on reader service card.

(Text begins on page 84)

```
B>PROLOG LOAD VALGOL
```

```
.begin
integer x;
0 =: x;
until x . = 15 .do
    .begin
edit (1+14*x-x*x, '+' );
print;
x+1 =: x
    .end
.end
```

```

** Compilation of "F,VAL" complete **
8. RT.

```

B>ASM P.BEZ

```
CP/M ASSEMBLER - VER 2.0
01F7
001H USE FACTOR
END OF ASSEMBLY
```

B>LOAD P

```
FIRST ADDRESS 0100
LAST ADDRESS 01F6
BYTES READ 00B9
RECORDS WRITTEN 02
```

 $E \supset F$

type p.val

```
.begin
integer x ;
0 =: x ;
until x ,= 9 .do
  .begin
    edit ( x*x + 1 , '+' ) ;
    print ;
    x + 1 =: x
  .end
.end
```

Adp

- **The THUNDER C Compiler** - Operates under the APPLE Pascal 1.1 or 1.2 operating system or ProDOS. Create fast native 6502 programs to run as stand alone programs or as subroutines to Pascal programs. A major subset of the C defined by K & R. Includes a users guide, newsletters, Macro preprocessor, runs on APPLE II, II+, //e, //c. Source code for libraries is included. **Only \$49.95**

• **ASSYST: The Assembler System** - A complete 6502 editor/ assembler and lister for APPLE DOS 3.3 or ProDOS. Menu driven, excellent error trapping, users guide, demo programs, source code for all programs! Great for beginners. **Only \$29.95**

- **THUNDER XREF** - A cross reference utility for APPLE Pascal 1.1. XREF generates cross references for each procedure. **Only \$19.95**

• **PDL - Pascal Development Library.** SCREENIO, DISKIO, CONVERSIONS, GRAPHICS, PLOT, MISC Units for Pascal programming. A must. **Only \$29.95**

- **PROPLOT** - Easy and powerful graphing package for plotting data. **Only \$24.95**

- **LINKIT** - Structured APPLESOFT processor. No more line numbers. **Only \$39.95**

POB 31501 Houston Tx. 77231

713-728-5501

Visa, Mastercard, COD, checks accepted

Add \$3.00 for P&H

Circle no. 56 on reader service card

Also Available for IBM PC
INCLUDES:

INCLUDES:

- ★ Cross Assembler ★
★ Cross Linker ★
★ Debugger ★
★ N.S. ISE Support ★
★ Librarian ★
★ Pascal Cross Compiler ★
★ C Cross Compiler ★

U.S. prices start at \$500

1283 Mt. View-Alviso Rd.

Suite B

Sunnyvale, Calif. 94089

408/745-7818 * TLX 4994264

Circle no. 68 on reader service card.

End Listing One

(Listing Two begins on next page)

Listing Two

VALGOL.LOG

```
?(( /*
  VALGOL.LOG
  Compiler for VALGOL I -- ver. 1.2

  Written in Micro-PROLOG.
  (Tested with Micro-PROLOG version 3.1 for CP/M-80)
  Translates VALGOL I to ASM-compatible 8080 assembly language.
  Uses file SEEKs for backtracking.

      written by: G. A. EDGAR      status: public domain

  0.3      September 9, 1984      M80, Z80
  1.0      September 16, 1984     output for ASM
  1.1      November 1, 1984       display input
  1.2      November 10, 1984      version for DDJ

  Usage - in SIMPLE syntax:
    &.is("FOO.VAL" compile "FOO.ASM")
    - in standard syntax:
    &.?((compile "FOO.VAL" "FOO.ASM"))
))

?(( /* ----- Language independent part -----))

?(( /* compile: input X, output Y))
((compile X Y)
  (message x) (P x) (PP) (P X "->" Y) (PP) (PP)
  (KILL labelused) (ADDCL ((labelused 0)) )
  (KILL last-shown) (ADDCL ((last-shown (-1|-1))) )
  (OPEN X) (KILL infile) (ADDCL ((infile X)) )
  (erase Y) (CREATE Y) (KILL outfile) (ADDCL ((outfile Y)) )
  (skip ((0|0)|(0|0)) Z1 ) (syntax z) (Q Z1 Z z)
  (CLOSE X) (CLOSE Y)
  (PP) (PP ** Compilation of X complete **)
((compile X Y)
  (PP) (PP ** Syntax error **)
  (CLOSE X) (CLOSE Y))
((C X Y)
  (compile X Y))

?(( /* Q: find it in the language-specific database))
((Q X Y Z)
  (Z | X1) (sequential X Y | X1))
((sequential X X))
((sequential X Y (z|Z)|y)
  (z X X1|Z) / (sequential X1 Y|y))

?(( /* out: send a line to outfile))
((out (y1|z1) (y1|z2) | X)
  (outfile Z) (SEEK Z z1) (outx Z | X)
  (outx Z "^M^J") (SEEK Z z2))
((outx Z))
((outx Z X | x)
  (W Z (X)) / (outx Z | x))

?(( /* match: the input matches string Z))
((match X Y Z)
  (STRINGOF Z1 Z) / (matchx X X2 Z1) (skip X2 Y))
((matchx X Y (x|Z))
  (inchar X X1 x1) (EQ x x1) / (matchx X1 Y Z))
((matchx X X ()))

?(( /* empty: matches automatically))
((empty X X))

?(( /* multiple: match the following zero or more times))
((multiple X Y|Z)
  (once X X1|Z) / (multiple X1 Y|Z))
((multiple X X|Z))
((once X X))
((once X Y (z|Z)|y)
  (z X X1|Z) / (once X1 Y|y))

?(( /* label: generate a new label))
((label X X Y)
  (labelused Y1) (SUM Y1 1 Y) (KILL labelused)
  (ADDCL ((labelused Y)) ))
```

(Continued on page 92)

Poor Person Software
Introduces

Write-Hand-Man

Desk accessories for CP/M

Write-Hand-Man lets you take notes, check phone numbers, make appointments, and countless other tasks without leaving Wordstar, dBase, Multiplan, or any other application. Enter **Write-Hand-Man** with a single key-stroke and choose the program you want. When you leave **Write-Hand-Man**, your application continues normally.

\$49.95 plus tax delivers **Write-Hand-Man** and 4 companion programs; **Notepad**, **Phonebook**, **Calendar**, and **Termcomm**. User written programs are easily added. All you need is M80 or some other LINK-80 compatible assembler.

Other CP/M products available from **Poor Person Software**: **Poor Person's Spooler** (\$49.95), **Poor Person's Spelling Checker** (\$29.95), **Poor Person's Spread Sheet** (\$29.95), **Keyed Sequential Files** (\$39.95), **Poor Person's Menus** (\$29.95), **aMAZEing Game** (\$29.95), **Window System** (\$29.95), **Crossword Game** (\$39.95), **Mailing Label Processor** (\$29.95). Shipping included.

All products available on IBM 8 inch and Northstar 5 inch disks. Other 5 inch formats add \$5 handling charge. No credit cards.

Poor Person Software

3721 Starr King Circle
Palo Alto, CA 94306
tel 415-493-3735

CP/M is a registered trademark of Digital Research

Circle no. 71 on reader service card.

WIZARD C

Fast compiles, fast code and great diagnostics make **Wizard C** unbeatable on MSDOS. Discover the powers of **Wizard C**:

- ALL UNIX SYSTEM III LANGUAGE FEATURES.
- UP TO A MEGABYTE OF CODE OR DATA.
- SUPPORT FOR 8087 AND 80186.
- FULL LIBRARY SOURCE CODE, OVER 200 FUNCTIONS.
- CROSS-FILE CHECKS OF PARAMETER PASSING.
- USES MSDOS LINK OR PLINK-86.
- CAN CALL OR BE CALLED BY PASCAL ROUTINES.
- IN-LINE ASSEMBLY LANGUAGE.
- 240 PAGE MANUAL WITH INDEX.
- NO LICENSE FEE FOR COMPILED PROGRAMS.

The new standard for C Compilers on MSDOS!

Only \$450

For more information call (617) 641-2379

WSS

Wizard Systems Software, Inc.

11 Willow Ct., Arlington, MA 02174

Visa/Mastercard accepted

Circle no. 116 on reader service card.

Indexed File Manager

using

B-Trees

\$75.00

+ 2.00 Postage
source included

C Programmers, We provide the record handling that C left out.

The **softfocus** BTree library is a record oriented function package that uses balanced BTree indexing for guaranteed fast access. Add our functions to your C library and greatly reduce application development time.

- **High speed** keyed and sequential file handling. Up to 16.7 million records per file.
- Source code supplied; conforms to **K&R standard** to ensure portability.
- **No royalties** on application programs.
- Documentation and **example programs** included to help you use BTrees.
- **Full feature** product at a fraction of the cost of competing BTree software.

Join the growing number of satisfied programmers using **softfocus** BTrees.

To order call

softfocus

Credit cards accepted.

1277 Pallatine Drive
Oakville, Ontario L6H 1Z1
(416) 844-2610

Dealer Inquiries Invited.



Circle no. 89 on reader service card.

TURBO ASSEMBLER

Introducing **FAST ASSEM-86™**, the **TURBO PASCAL™** of IBM PC assemblers. **FAST ASSEM-86™ (FASM)** is significantly faster and easier to use than the IBM Macro-Assembler (MASM). Whether you are new to assembly language and want to quickly write a small assembly language routine, or are an experienced MASM user tired of waiting months to assemble large files, **FAST ASSEM-86** will bring the excitement back to assembly language.

FAST ASSEM-86 IS MUCH FASTER:

- How fast is **FASM™**? The graph below shows relative assembly times for a 48K source file. For large files like this we blow MASM's doors off at 3 times their speed. For smaller 8K files we positively vaporize them at 6 times their speed.

FASM™	(110 sec.)	
MASM	(340 sec.)	

- **FAST ASSEM-86** is faster for the following reasons: (1) Written entirely in assembly language (unlike MASM). (2) Editor, assembler and source file always in memory so you can go instantly from editing to assembling and back. (3) Eliminates the time needed to LINK programs. Executable .COM files can be created directly. (Also creates .OBJ files compatible with the IBM linker).

FAST ASSEM-86 IS EASIER TO USE:

FASM includes many other features to make your programming simpler.

- Listings are sent directly to screen or printer. Assemblies can be single stepped and examined without having to leave the editor.
- Access the built in cross reference utility from the editor.
- Full support of 186 and 286 (real mode) instructions.
- Both Microsoft and 8087 floating point formats are supported. 8087 and 287 instructions supported directly without macros for faster assembly.
- Calculator mode: Do math in any radix even using symbols from the symbol table.
- Direct to memory assembly feature lets you test execute your code from editor.
- Coming soon: A coordinated symbolic debugger.

COMPATIBILITY: **FASM** is source code compatible with MASM and supports macros, records and structures.

Introductory Price \$49
With .OBJ Capability \$99

Speedware™

IBM, Turbo Pascal, Microsoft trademarks of IBM Corp., Borland Intern., Microsoft Corp.

Dealer inquiries welcome

916-966-6247

Box D2, 2931 Northrop Avenue
Sacramento, CA 95825

Circle no. 97 on reader service card.

Compiler in Prolog (Listing Continued, text begins on page 84)

Listing Two

```

?(( /* string: match a string quoted within characters x))
((string X Y x Z)
  / (inchar X X2 x1) (EQ x x1) /
  (stringx X2 X3 x Z1) (skip X3 Y) (STRINGOF Z1 Z))
((stringx X Y x ())
  (inchar X Y x1) (EQ x x1) / )
((stringx X Y x (y | Z))
  (inchar X X1 y) / (stringx X1 Y x Z))

?(( /* id: match an identifier))
((id X Y Z)
  (idx X X2 Z1) (skip X2 Y) (STRINGOF Z1 Z))
((idx X Y (x|Z))
  (inchar X X1 x) (alphabetic x) / (alphanum X1 Y Z))
((alphanum X Y (x|Z))
  (inchar X X1 x) (alphanumeric x) / (alphanum X1 Y Z))
((alphanum X X ()))

?(( /* number: match a number))
((number X Y Z)
  / (numberx X X2 Z1)
  (NOT EQ Z1 ()) (skip X2 Y) (STRINGOF Z1 Z))
((numberx X Y (x|Z))
  (inchar X X1 x) (digit x) / (numberx X1 Y Z))
((numberx X X ()))

((alphanumeric x)
  (alphabetic x))
((alphanumeric x)
  (digit x))

((alphabetic x)
  (LESS "0" x)(LESS x "L"))
((alphabetic x)
  (LESS "" x)(LESS x "Z"))

((digit x)
  (LESS "/" x)(LESS x ";"))

?(( /* skip: skip over characters listed as "skipchar"))
((skip X Y)
  (inchar X X1 x) (skipchar x) / (skip X1 Y))
((skip X X))

((skipchar " ")
  (skipchar "~I")
  (skipchar "~J")
  (skipchar "~M"))

?(( /* inchar: input one character, show on console first time))
((inchar (y1|z1) (y2|z1) x)
  (infile Y) (SEEK Y y1) (RDCH Y x) (SEEK Y y2) (last-shown y3)
  (IF (before y3 y2)
    ((P x) (KILL last-shown) (ADDCL ((last-shown y2)) ))
    ()))

?(( /* before: compare two file positions ))
((before (y1|z1) (y1|z2))
  / (LESS z1 z2))
((before (y1|z1) (y2|z2))
  (LESS y1 y2))

?(( /* erase: erase a file, no failure))
((erase Y)
  (ERA Y) /)
((erase Y))

?(( /* --- VALGOL specifics ----- ))

((message "VALGOL 1 compiler - translates VALGOL to ASM"))
((syntax PROGRAM))

((PROGRAM
  (match ".begin")
    (out "VCPM EQU 0")
    (out "VBDOS EQU 5")
    (out "VTPA EQU 256")
    (out "VCR EQU 13")
    (out "VLF EQU 10")
    (out " ORG VTPA")
    (out " LXI SP,VSTACK")
  (Q OPT-DECLARATION)
  (Q STATEMENT)

```

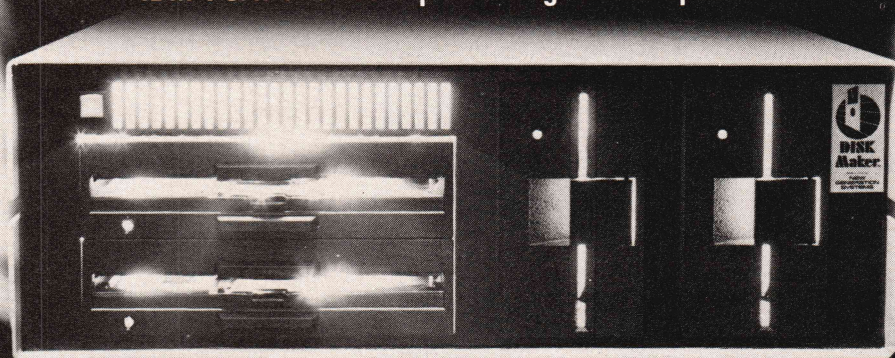
```
(multiple
  (match ";")
  (Q STATEMENT))
(match ".end")
```

```
(out " JMP VCFM")
(out "VMULT:")
(out " MOV B,H")
(out " MOV C,L")
(out " XRA A")
(out " MOV H,A")
(out " MOV L,A")
(out " MVI A,16")
(out "VMULT1: FUSH PSW")
(out " DAD H")
(out " XRA A")
(out " MOV A,C")
(out " RAL")
(out " MOV C,A")
(out " MOV A,B")
(out " RAL")
(out " MOV B,A")
(out " JNC VMULT2")
(out " DAD D")
(out "VMULT2: POP PSW")
(out " DCR A")
(out " ORA A")
(out " JNZ VMULT1")
(out " RET")
(out "VEDIT:")
(out " MOV A,H")
(out " ORA L")
(out " JZ VEDIT1")
(out " MVI A,'")
(out " CALL VCFMOUT")
(out " DCX H")
(out " JMP VEDIT")
(out "VEDIT1: POP H")
(out "VEDIT2: MOV A,M")
(out " CFI 0")
(out " INX H")
(out " JZ VEDIT3")
(out " CALL VCFMOUT")
```

(Continued on next page)

The Most Affordable Disk MakerTM NEW! in the Universe

Now with over 25 MSDOS formats, 3 1/2" formats,
IBM PCAT and word processing format options



Disk Maker II shown
with opt. drives

Download fast, read over 200 formats easily, reformat rapidly

The more disk formats you work with, the more our Disk MakerTM system saves time and money by reading and/or writing disks in any of over 200 formats. No modems, no patches, no other special software necessary.

Disk Maker II is a complete, stand alone system with one 8" DSDD disk drive, one 48 tpi 5 1/4" DSDD disk drive, 6 MHZ Z80B, 64K CP/M system with Disk MakerTM software. (96 tpi and second 8" drive optional.) Just plug in your terminal and make disks! Bundled software includes MicroShellTM/MCALL-II communications software. Base price: \$3,395.

Supported with comprehensive, easy-to-read manual, software updates (\$50.00, all formats in revision), and additional drives and hard disk options.

Disk MakerTM
prices from
\$1,695

Disk Maker I runs as a peripheral with an S-100 system and comes with S-100 controller board, one 48 tpi DSDD 5 1/4" disk drive, dual drive cabinet and power supply, cables and Disk Maker software. 96 tpi and 8" drives are optional. Base price: \$1,695.

**NEW
GENERATION
SYSTEMS**

1800 Michael Faraday Drive, Suite 206, Reston, VA 22090
(703) 471-5598 Order Line: (800) 368-3359 Dealer inquiries welcomed.

Circle no. 76 on reader service card.

Listing Two

```

(out      "      JMP      VEDIT2")
(out      "VEDIT3:  PUSH    H")
(out      "      RET")
(out      "VPRINT:")
(out      "      MVI      A,VCR")
(out      "      CALL     VCPMOUT")
(out      "      MVI      A,VLF")
(out      "      CALL     VCPMOUT")
(out      "      RET")
(out      "VCPMOUT:")
(out      "      PUSH     H")
(out      "      MOV      E,A")
(out      "      MVI      C,2")
(out      "      CALL     VBDOS")
(out      "      POP      H")
(out      "      RET")
(out      "      DS       60")
(out      "VSTACK:  DW      0")
(out      "      END") ))

```

```

((OPT-DECLARATION
  (Q DECLARATION)
  (match ";")))
((OPT-DECLARATION
  (empty)))

```

```

((DECLARATION
  (match ".integer")
  (Q ID-SEQUENCE)
  (label X1)
  (out      "      JMP      V" X1)
  (out      "V" X1 " :") ))

```

```

((ID-SEQUENCE
  (Q IDENTIFIER)
  (multiple
    (match ",")
    (Q IDENTIFIER) )))

```

```

((IDENTIFIER
  (id Z)
  (out      Z "V:    DS      2") ))

```

```

((STATEMENT
  (Q IO-STATEMENT)))
((STATEMENT
  (Q ASSIGNMENT-STATEMENT)))
((STATEMENT
  (Q UNTIL-STATEMENT)))
((STATEMENT
  (Q CONDITIONAL-STATEMENT)))
((STATEMENT
  (Q BLOCK)))

```

```

((BLOCK
  (match ".begin")
  (Q DECL-OR-ST)
  (multiple
    (match ";")
    (Q STATEMENT) )
  (match ".end") ))

```

```

((BLOCK
  (match ".begin")
  (match ".end") ))

```

```

((DECL-OR-ST
  (Q DECLARATION)))
((DECL-OR-ST
  (Q STATEMENT)))

```

```

((IO-STATEMENT
  (match "edit")
  (match "(")
  (Q EXPRESSION)
  (match ",")
  (string "" Z)
  (match ")") ))

```

```

(out      "      CALL     VEDIT")
(out      "      DB       ' Z ',0")

```

```

((IO-STATEMENT
  (match "print")
  (out      "      CALL     VPRINT") ))

```

```

((CONDITIONAL-STATEMENT
  (match ".if")
  (Q EXPRESSION)
  (match ".then")
  (label X1) (label X2)
  (out      "      MOV      A,H")

```

```

(out      "      ORA      L")
(out      "      JZ       V" X1)

(Q STATEMENT)
(match ".else")
(out      "      JMP      V" X2)
(out      "V" X1 ";;")
(out      "V" X2 ";;" ))

((UNTIL-STATEMENT
 (match ".until")
 (Q EXPRESSION)
 (match ".do")
 (Q STATEMENT)
 (out      "      MOV      A,H")
 (out      "      ORA      L")
 (out      "      JNZ      V" X2)
 (out      "      JMP      V" X1)
 (out      "V" X2 ";;" ))

((ASSIGNMENT-STATEMENT
 (Q EXPRESSION)
 (match "=:")
 (id Z)
 (out      "      SHLD      " Z "V" ))

((EXPRESSION
 (Q EXPRESSION1)
 (Q OPT-RIGHT-SIDE)))

((OPT-RIGHT-SIDE
 (match ".*=")
 (Q EXPRESSION1)
 (label X1) (label X2)
 (out      "      PUSH      H")
 (out      "      POP       D")
 (out      "      MOV      A,L")
 (out      "      SUB       E")
 (out      "      JNZ      V" X2)
 (out      "      MOV      A,H")
 (out      "      SBB       D")
 (out      "      JNZ      V" X2)
 (out      "      LXI      H,1")
 (out      "      JMP      V" X1)
 (out      "V" X2 ";;")
 (out      "      LXI      H,0")
 (out      "V" X1 ";;" ))

((OPT-RIGHT-SIDE
 (empty)))

((EXPRESSION1
 (Q TERM)
 (multiple
 (Q SIGNED-TERM))))

((SIGNED-TERM
 (match "+")
 (Q TERM)
 (out      "      PUSH      H")
 (out      "      POP       D")
 (out      "      DAD       D"))

((SIGNED-TERM
 (match "-")
 (Q TERM)
 (out      "      PUSH      H")
 (out      "      POP       D")
 (out      "      MOV      A,E")
 (out      "      SUB       L")
 (out      "      MOV      L,A")
 (out      "      MOV      A,D")
 (out      "      SBB       H")
 (out      "      MOV      H,A" ))

((TERM
 (Q PRIMARY)
 (multiple
 (match "x")
 (Q PRIMARY)
 (out      "      PUSH      H")
 (out      "      POP       D")
 (out      "      CALL      VMULT" ))))

((PRIMARY
 (id Z)
 (out      "      LHLD      " Z "V"))

((PRIMARY
 (number Z)
 (out      "      LXI      H," Z )))

((PRIMARY
 (match "(")
 (Q EXPRESSION)
 (match ")") ))

```

End Listing Two

(Listing Three begins on next page)

Compiler in Prolog

Listing Three

(Listing Continued, text begins on page 84)

Part of VALGOL.LOG in SIMPLE syntax

```
X compile Y if
  message (Z) and
  P (Z) and
  PP and
  PP and
  KILL (labelused) and
  add ((0 labelused)) and
  KILL (last-shown) and
  add (((-1|-1) last-shown)) and
  OPEN (X) and
  KILL (infile) and
  add ((X infile)) and
  erase (Y) and
  CREATE (Y) and
  KILL (outfile) and
  add ((Y outfile)) and
  skip (((0|0) 0|0) x) and
  syntax (y) and
  QQ (Q x z y) and
  CLOSE (X) and
  CLOSE (Y) and
  PP and
  PP (***) Compilation of X complete **)
X compile Y if
  PP and
  PP (** Syntax error **) and
  CLOSE (X) and
  CLOSE (Y)
X alphanumeric if
  X alphabetic
X alphanumeric if
  X digit
X alphabetic if
  @ LESS X and
  X LESS C
X alphabetic if
  ^ LESS X and
  X LESS C
X digit if
  / LESS X and
  X LESS :
```

End Listing Three

Listing Four

P.ASM

```
VCPM EQU 0
VBDOS EQU 5
VTPA EQU 256
VCR EQU 13
VLF EQU 10
ORG VTPA
LXI SP,VSTACK
JMP V1
xV: DS 2
V1:
LXI H,0
SHLD xV
V2:
LHLD xV
PUSH H
LXI H,15
POP D
MOV A,L
SUB E
JNZ V5
MOV A,H
SBB D
JNZ V5
LXI H,1
JMP V4
V5:
LXI H,0
V4:
MOV A,H
ORA L
JNZ V3
LXI H,1
PUSH H
```

```
LXI H,14
PUSH H
LHLD xV
POP D
CALL VMULT
POP D
DAD D
PUSH H
LHLD xV
PUSH H
LHLD xV
POP D
CALL VMULT
POP D
MOV A,E
SUB L
MOV L,A
MOV A,D
SBB H
MOV H,A
CALL VEDIT
DB '+',0
CALL VPRINT
LHLD xV
PUSH H
LXI H,1
POP D
DAD D
SHLD xV
JMP V2
V3:
JMP VCPM
VMULT:
MOV B,H
MOV C,L
XRA A
MOV H,A
MOV L,A
MVI A,16
VMULT1: PUSH PSW
DAD H
XRA A
MOV A,C
RAL
MOV C,A
MOV A,B
RAL
MOV B,A
JNC VMULT2
DAD D
VMULT2: POP PSW
DCR A
ORA A
JNZ VMULT1
RET
VEDIT:
MOV A,H
ORA L
JZ VEDIT1
MVI A,' '
CALL VCPMOUT
DCX H
JMP VEDIT
VEDIT1: POP H
VEDIT2: MOV A,M
CPI 0
INX H
JZ VEDIT3
CALL VCPMOUT
JMP VEDIT2
VEDIT3: PUSH H
RET
VPRINT:
MVI A,VCR
CALL VCPMOUT
MVI A,VLF
CALL VCPMOUT
RET
VCPMOUT:
PUSH H
MOV E,A
MVI C,2
CALL VBDOS
POP H
RET
DS 60
VSTACK: DW 0
END
```

End Listings

DDJ Classifieds

Software

PCBTAM

Communications Access Method

Allows an IBM PC or compatible to perform BISYNC communication. PCBTAM is a general purpose interrupt driven access method usable by any Microsoft language. Requires IBM BSCA card and PC-DOS.

- Source or object license
- Asynchronous version available (PCATAM)
- Modifiable for other USARTS.

For details write:

SYMBIOTIC

Symbiotic Protocol Converters, Inc.
1011 Clifton Avenue, Clifton, New Jersey 07013
(201) 777-8454

Olive Branch Software COPY PROTECTION

SLK/F places an assembled or compiled program on a diskette with 4 different copy-resistant features in such a way that it runs normally, but cannot be copied by backup programs such as COPYPC. The rest of the diskette is available as normal, and DOS may be added. Price \$150. Olive Branch Software
1715 Olive Street
Santa Barbara, CA 93101
(805)569-1682

DriveLiner

Drive Alignment Test
Program for CP/M 2.2/3.1
With Dysan 8" SSD
Diagnostic Disk
\$65 check/MO postpaid
Chandler Software, 273 W
Shore Dr, Marblehead MA
01945 (617)631-4685

BRINGING YOUR PROGRAMS OUT OF THE CLOSET?

DISKETTES & DUPLICATION

Large or Small - Let's Talk!!

5 1/4" DISKETTES

SS/DD **99¢** EA. DS/DD **\$1.10** EA.

Minimum order 250 disks.

8" - Color Jackets - 3 1/2" CALL!!

*Hundreds of professionals
can't be wrong!!!*

**CALL
TODAY!!** 800-437-0500
800-435-7400 In CA
(408) 262-3008



780 Trimble Rd. • Suite 608
San Jose, CA 95131

TECMAR GRAPHICS LIBRARY

TECMAR lets you do high-res graphics on your TECMAR Graphics Master. Features windowing, viewporting, clipping, axis rotation. Similar to Tektronix graphics. Includes screen dump/restore, Epson screen print, support for HP and Western Graphtec plotters. Includes three curve-fitting programs and graphics application SOURCE CODE. Requires MS-FORT 3.20, or Lahey F77L. Price: \$195.

ADVANCED SYSTEMS
CONSULTANTS
18653 Ventura Boulevard,
Suite 351
Tarzana, California 91356
(818) 990-4942

Offering low cost
advertising reaching
thousands of computer
programmers and
professionals each
month. Departments to
choose from include:
Software
Hardware
Accessories/Supplies
Career Opportunities
Services
User Groups. Special
categories are also
available.

Dr. Dobb's Journal is pleased to announce the DDJ Classifieds

RATES: DISPLAY ADVERTISERS: Price per column inch \$100. Ad must run in 3 consecutive issues.

LINE ADVERTISERS: Price per line \$12 (35 characters per line including spaces). Minimum of 5 lines. 3 consecutive issues. Add \$3 per line for boldface type. Add \$25 if a background screen is desired.

DISCOUNTS: Frequency discounts for 6 consecutive ads (less 7%) and for 12 consecutive ads (less 15%).

MECHANICAL REQUIREMENTS: Camera ready art or typewritten copy only (phone orders excepted). Display: Specify desired size, include \$20 if reduction is required. Line: Specify characters in boldface, caps. Allow 6 weeks for publication.

PAYMENTS: Full payment in advance. Check, money order, Visa, M/C, AmEx.

Run this ad in _____ issues

Size: _____ col x _____ inches or 1 col x _____ lines

Payment by: _____ check _____ m/o _____ Visa _____ M/C _____ AmEx

Card # _____ Exp Date _____

Signature _____

Print Name _____

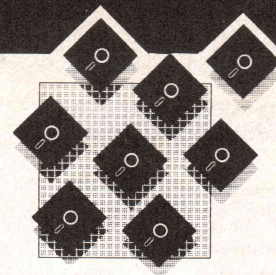
Company _____

Address _____

City _____ St _____ Zip _____

Phone _____ Current Subscriber _____

**Mail to or phone Alex Williams, DDJ Classifieds, 2464 Embarcadero
Way, Palo Alto, CA 94303 (415) 424-0600**



Toolworks C/80 and Toolworks C/80 Mathpack, Version 3.1

Company: Software Toolworks,
15233 Ventura Blvd., Suite
1118, Sherman Oaks, CA
91403

Computer: Heath; Kaypro II, 4,
and 10; Osborne 1 and the
Executive; DEC VT-180 and
Rainbow; Xerox 820

Price: Compiler, \$49.95;
Mathpack, \$29.95

Circle Reader Service No. 101
Reviewed by D. C. Shoemaker

Few readers of *Dr. Dobb's Journal* are unaware of the contribution made to small computer system programmers by Ron Cain and his minimal C subset compiler. Over the years since its introduction, there have been many extensions and improvements, all building on his basic idea. One of the best and most complete versions is from Walt Bilofsky's Software Toolworks. First released in 1981, Version 3.1 is the most current and comes the closest to being a complete C compiler.

This software is a bargain at the price. The package comes in two parts, sold separately: the C compiler itself is \$49.95, and the C/80 Mathpack is \$29.95. There's a reason for the way the software is packaged and priced. Those just learning C probably have no need for the FLOAT and LONG data types that the Mathpack provides. Adding these features somewhat increases the size of the libraries and the assembled code. If you need them, Mathpack is a bargain-level entry into other more powerful capabilities given by the additional data types. Therefore, you don't have to pay for something you don't need, but you aren't limited to

an entry-level compiler that can't do serious work.

C/80 is a complete implementation of the C language described by Brian Kernighan and Dennis Ritchie in *The C Programming Language* (Prentice-Hall, 1978) with the exception of the following features:

- FLOAT and LONG data types (available in the C/80 Mathpack)
- DOUBLE data type
- typedef
- Arguments to #define macros
- #line
- Declarations within nested blocks
- Bit fields

Whether or not these omissions are significant depends a great deal on what you want to do with the software. If you want to learn the language and get a whiff of programming in C, the C/80 package is sufficient. If you want to write some fast-running utility software and don't want to invest the time in assembly language, these omissions will cause no problems.

C/80 supports structures, statics, initialization, casts, compile time evaluation of constant expressions, and all the other C language features. It's worth mentioning some of the features that C/80 does provide, however, because they're often lacking in other subsets:

- Unix-style I/O redirection
- Conventional C I/O and string library
- Formatted and random access file I/O
- Dynamic storage allocation
- Runtime execution profile capability
- In-line assembly language capability

In addition, C/80 can generate source code compatible with Macro-80 and RMAC. Should you have neither, a serviceable assembler comes with the C/80 package. It works well and does the job with no fuss.

If you have a version of C/80 earlier than 3.0, here are the main changes in version 3.1:

- An expanded runtime library
- ROMable code
- RMAC compatibility
- A menu-configurable compiler
- \ at the end of a line for continuation
- #ifneed for selective compilation
- True alloc/free
- Command line wildcard expansion
- CP/M files now written in 128-byte records
- File 0 always the terminal

The changes from version 3.0 to 3.1 are to correct minor bugs and speed code generation. Some new compiler switches have been added, and you are now prohibited from reading from and writing to the same open file. If you have one of the earlier versions of C/80, return the original disk to Software Toolworks with \$10.00, and they will give you the most recent version. I bought my first copy of C/80 in 1981 and have updated it twice.

Software Toolworks makes the C/80 package available in a variety of formats. Walt Bilofsky was one of the first to support Heath's in-house disk operating system HDOS, and he still does. However, the standard remains CP/M, and Software Toolworks provides disks for Heath, both hard and soft sector; Kaypro II, 4, and 10; Osborne I and the Osborne Executive; DEC's VT-180 and the Rainbow; the Xerox 820; and the standard 8-inch CP/M single-density format. There's

even a version for the Epson QX-10. Currently no version is offered for the IBM PC, PCjr, or any MSDOS, PC DOS, or ZDOS computer because the programs are written in 8080 code, but I expect that will change. Incidentally, I'm using the CP/M version on a Heath H120 running CP/M-85 with no trouble at all. Just order the soft-sector Heath version.

Using C/80 is simple. The hard part (as usual) is writing the original C language program. Then you call C/80 and let it compile the source into 8080 assembly language code. Let's use one of the test files on the distribution disk, HELLO.C. We do this by typing "C80 HELLO." The output file from C/80 can be modified if you wish. I'm not the greatest assembly language programmer, so I don't change the code, but someone more experienced might be able to tighten it up in some areas.

Next, call the assembler of your choice. I use the one that comes with the package. It's invoked as you might expect, by typing "AS HELLO." When the assembly is finished, you have an executable file called HELLO.COM. It will be rather large, due to the inclusion of pieces from the I/O and other libraries, but that's part of the tradeoff. The completed code executes relatively quickly. I don't have the capability to do much comparative benchmarking, but comparisons I've seen suggest that in some cases C/80 is a bit faster than BDS C and in others a bit slower. On balance they seem about equal, except that BDS C has some nonstandard features that get in the way of portability, and it costs twice what C/80 costs. BDS C doesn't provide as many formats, either.

Software Toolworks doesn't charge a license fee for the COM files created by C/80. This means that you're free to market whatever you write and compile as you see fit. Unlike some of the so-called "big boys" of the software industry, Bilofsky encourages programmers to market software using his tools.

A word about documentation: If you're a first-time C user, the manual that accompanies C/80 won't be too helpful. There are some examples in

the documentation that you can play with, and several source code files on the distribution disks, but that's it for introductory material. Several good books on the market can help you get a better grasp of the language. The documentation lists eight of the most common of these, and new books appear almost monthly.

The C/80 manual is 50 pages long with a three-page index. It has 16 chapters, with most of the attention paid to a language summary, a description of the I/O and runtime libraries, and a discussion on running the compiler. The manual includes a brief discussion of the internal features of the compiler and useful tricks to bend the compiler to your will. There is also a detailed list of compiler error messages, what they mean, and what to do about them.

The C/80 Mathpack requires C/80 version 3.0 or greater. If you've been working with an earlier version of C/80, the Mathpack is a good reason to take advantage of the Software Toolworks' update policy. Mathpack comes on one disk with a short manual. The Mathpack adds true 32-bit LONG and FLOAT data types to the C/80 compiler and runtime library. It includes a program to patch the C/80 compiler to recognize LONG and FLOAT, a runtime library to perform 32-bit arithmetic, routines to convert between ASCII and floating-point representation, an augmented printf, and a transcendental library written in C.

In practical terms, this gives the C/80 a 32-bit signed number capability in the range -2,147,483,648 to 2,147,483,647. Floating-point numbers have 24-bit precision, which equates to about seven decimal digits, and an exponent range of 10 to the plus or minus 38th power.

Installing the Mathpack requires only that you run the modifying program CCONFIG.COM. A menu will appear with a range of options. Choose option N and press RETURN. This changes FLOAT and LONG from not available to available. Now type a Y in response to the question about making changes permanent, and you're done. Everything else about C/80 remains as before, except

NEW! for IBM® PC, XT, Compaq®, Corona®

FlipKit™

remembers the function keys...



...of all your favorite software. You select your own combination from 20 color coded function key overlays, and attach your FlipKit to your keyboard—that's all there is to it! Of course, FlipKit is non-glare, easily removable, and will not mark your keyboard. Every FlipKit includes preprinted overlays for:

Lotus™ 1-2-3™
Multiplan™ for IBM®
Framework™
Symphony™
dBase® III
IBM® Filing/
Reporting Assistant
Thinktank™
IBM® Writing Assistant
Microsoft® Word

Multimate™
Pfs:write
Wordstar®
Crosstalk XVI™
MS™ DOS

PLUS
5 "do-it-yourself"
erasable overlays
and keyboard
attachments

Only \$14.95! Order Now!

Visa, Mastercard only, call 24 hrs/7 days:

1-800-228-2028 ext. 120 toll free Cont. U.S.

In Nebraska, call 1-800-642-8300 ext. 120
Information, inquiries, call 415-325-8000

Or send check, m.o., credit card info. (#, exp. date, and signature) to:

Micro Direct International
P.O. Box 60987, Palo Alto, CA 94306

Include quantity, name, address, city, state, zip, phone. Sorry no CODs. Order several—shipping/handling per order: USA \$3.50/Canada \$6/Other countries \$15. CA residents add appropriate sales tax. Allow 3-4 weeks delivery (personal checks add 2 weeks). Dealer and corporate inquiries welcome.

FlipKit — Micro Direct International/Lotus 1-2-3, Symphony — Lotus Development Corp./Multiplan, MS Word, MS DOS — MicroSoft Corp./pfs:write — Software Publishing Corp./Framework, dBase III — Ashton Tate/IBM Writing Assistant, Filing Assistant, Reporting Assistant — International Business Machines Corp./MultiMate — MultiMate International Corp./WordStar — MicroPro International Corp./Crosstalk XVI — Microstuf, Inc./Thinktank — Living Videotext, Inc.

Circle no. 30 on reader service card.

that the size of the compiled and assembled code increases a bit.

The Mathpack manual is only 10 pages long with six chapters; it spends most of its time on installation and additions to the function library. You'll have to know how to make use of FLOAT and LONG before the Mathpack will do you much good.

In summary, if you want to learn C or if you already know a bit about the language and want an inexpensive but powerful implementation for your CP/M computer, my first choice would be the Software Toolworks C/80 and an introductory book. If you want an enhanced 32-bit capable C, add the Mathpack. For under \$80.00 you can't miss.

Disk Maker I

Company: New Generation Systems, 1800 Michael Faraday Drive, Suite 206, Reston, VA 22090
(703) 471-5598,
(800) 368-3359

Operating System: CP/M 2.2

Price: \$1695.00

Circle Reader Service No. 103

Reviewed by Jim Kronman

Disk Maker I from New Generation Systems offers a practical solution to the virtual Tower of Babel that exists in the world of CP/M and MSDOS disk formats. Acting as a peripheral device in an S-100 system, Disk Maker I can format, read, write, and duplicate over 170 disk formats (at the time of this review—the total grows every month or two), allowing you to copy virtually any soft-sectored format and transfer disk files between various CP/M and MSDOS disk formats. An optional file transfer utility program transfers files from DecMate II and Wang OIS word processing disk formats.

The standard Disk Maker I package consists of an S-100 disk controller board capable of supporting up to four disk drives, a 48 track-per-inch (tpi), double-sided 5¼-inch disk drive with power supply in a fan-cooled cabinet, and the Disk Maker software. Other disk drives for Disk Maker are a 96 tpi 5¼-inch drive, a

135 tpi 3½-inch drive, or a 96 tpi 1.2 Mb IBM PC/AT drive (as used in the IBM PC/AT) installed in the cabinet with the 48 tpi drive. A double-sided double-density (DSDD) 8-inch drive with its power supply in a separate cabinet is also available. Software options are the word processing transfer utilities and disk drive test software.

If you don't have an S-100 system, do not stop reading: New Generation Systems also offers a stand-alone Disk Maker II system. It is a 6 MHz, Z80, 64K CP/M 2.2 system that includes one DSDD 8-inch drive and one 48 tpi, DSDD 5¼-inch drive plus the same software provided with Disk Maker I. You can add the same options described for Disk Maker I, up to a total of four drives; a 10 Mb hard disk option is also available. Having reviewed the manual for the Disk Maker II (but not having used the equipment), I am reasonably confident that my remarks about Disk Maker I will be applicable to the operation of Disk Maker II as well.

The Disk Maker I controller uses the Western Digital 1795 controller chip, allowing the software to read or write disks in all 170-plus formats shown in the Table on page 101 (if you have the appropriate drives, of course). Disk Maker I cannot accommodate some disk formats such as Apple, NorthStar, Micropolis, Commodore, and Victor; these formats either are hard-sectored or depend on special tricks in the disk controller hardware.

When you have the optional 96 tpi 5¼-inch drive installed, the software allows you to set the 96 tpi drive as a read-only drive for a 48 tpi format; this allows you to transfer files from one 5¼-inch disk to another without having to do an intermediate copy. With the two 5¼-inch drives set to the same format, rapid disk duplication is possible using the DUPE utility program. New formats are added regularly, and New Generation Systems offers each upgrade to current users for only \$50.

Programs

The Disk Maker I software includes these programs:

DMINSTAL.COM—sets user-deter-

mined and system-dependent options
DMSET.COM—loads Disk Maker BIOS into the system

DMFORM.COM—formats disks

WHATDISK.COM—determines exact format of IBM, Z-100, or Morrow disk

MC.COM—Mass Copy, a file copy program

DUPE.COM—rapid track-for-track copy program with optional verify

DMCOMP.COM—rapid track-for-track compare program

TOMS.COM—CP/M < = > PCDOS file exchange utility

Various public domain utilities such as SWEEP and D normally are provided on the distribution disk as space permits. Two or more versions of DMSET and DMFORM are provided to accommodate various hardware peculiarities.

The optional software is:

DDD.COM—disk alignment program used with special disks

DEC2CPM—utility to transfer files from DecMate II word processor to CP/M

WANG2CPM—utility to transfer files from Wang OIS word processor to CP/M

Requirements

Disk Maker I is designed to operate under standard CP/M 2.2 with an 8080, 8085, or Z80 CPU. New Generation Systems recommends using it with a standard CP/M system only, but I run it with CP/M 2.2 and ZCPR, version 1, without any problems. The controller board and Disk Maker software normally use ports 98 to 9F (hex), but New Generation Systems provides software and hardware modification instructions to relocate the ports to other addresses if you have an unresolvable conflict in your system.

Although a 64K system is recommended for Disk Maker I, it will run in as little as 48K. The more disks you attach to the Disk Maker, the more memory the software requires. When you use the TOMS utility, a smaller TPA means that you may not be able to handle a full directory on an MSDOS disk; my 57K system will handle the maximum. Double-density

Apricot CP/M-86	3"(SSDD)	306K	Intertec HeadStart 3"(SSDD)	372K
Access	(SSDD)	169K	Actrix Matrix	(DSDD) 348K
AMPRO	(SSDD)	188K	AMPRO	(DSDD) 386K
Balcones BRAVO	(DSDD)	346K	Bondwell Model 14/16	(DSDD) 346K
BYAD	(SSDD)	154K	Casio FP 1000/1100	(DSDD) 300K
Compuview CP/M-86	(SSDD)	193K	Cromemco	(SSSD) 81K
Cromemco	(SSDD)	188K	Cromemco C-10	(DSDD) 386K
Davidge 1024	(DSDD)	368K	DEC VT180	(SSDD) 169K
Digilog	(DSDD)	332K	Eagle I	(SSSD) 70K
Epson QX-10 CP/M	(DSDD)	378K	Epson QX-10 MF	(DSDD) 278K
Fujitsu Micro 16s	(DSDD)	314K	Heath H-37	(SSDD) 148K
Heath H-37	(DSDD)	304K	Heath w/CDR Systems	(DSDD) 380K
Heath w/Magnolia	(SSDD)	162K	HP 86/87/125	(DSDD) 248K
IBM PC (CP/M-86)	(SSDD)	154K	IBM PC (CP/M-86)	(DSDD) 314K
ICL PC1 10,30,31,32	(DSDD)	254K	Insight Dev. IQ-120	(SSDD) 146K
KayPro II	(SSDD)	191K	KayPro 4/10	(DSDD) 394K
Lobo Max-80	(SSDD)	164K	Lobo Max-80 35tk	(SSDD) 142K
Lobo Max-80 35tk	(DSDD)	296K	Microbee	(DSDD) 171K
Microbee	(DSDD)	386K	Molecular 9X	(DSDD) 338K
Morrow Micro Dec I/II	(SSDD)	186K	Morrow Micro Dec III	(DSDD) 384K
Morrow Micro Dec II	(DSDD)	384K	MultiTech MIC-501	(SSDD) 168K
NCR Decision Mate V	(DSDD)	304K	NEC PC-8000 Series	(SSDD) 169K
NEC PC-8000 Series	(DSDD)	300K	Osborne	(SSSD) 90K
Osborne DD/Executive	(SSDD)	183K	Otrona Attache	(DSDD) 360K
PMC 101 (CP/M 3.0)	(DSDD)	386K	Sanyo 1000/1100/1150	(DSDD) 310K
SD Sales	(SSSD)	71K	SD Sales	(SSDD) 112K
Sharp 3540	(DSDD)	292K	SuperBrain	(SSDD) 162K
SuperBrain	(DSDD)	340K	Systel III	(DSDD) 346K
TeleVideo 802/3/6	(DSDD)	340K	TeleVid. 806 TurboDos	(DSDD) 396K
TI Professional	(SSDD)	154K	Toshiba T100	(DSDD) 254K
TRS-80 Mod I Omikron	(SSSD)	70K	TRS-80 Mod I Omikron	(SSDD) 131K
TRS-80 Mod III MM	(SSDD)	186K	TRS-80 Mod IV	(SSDD) 154K
TRS-80 IV Montezuma	(SSDD)	166K	Versa Floppy II	(DSDD) 278K
Xerox 820	(SSSD)	82K	Xerox 820	(SSDD) 155K
Zenith Z-100	(SSDD)	148K	Zenith Z-100	(DSDD) 304K
Zenith Z-100 (orig)	(DSDD)	304K	Zorba	(DSDD) 388K
ALTOS 586	96(DSDD)	700K	AMPRO	96(SSDD) 386K
AMPRO	96(DSDD)	782K	Analyzer	96(DSDD) 620K
Archives I	96(SSDD)	386K	Archives II	96(DSDD) 786K
Associate +	96(DSDD)	786K	CompuPro 10	96(DSDD) 772K
CompuStar 40	96(DSDD)	786K	DEC Rainbow	96(SSDD) 386K
Digilog DBS 16	96(DSDD)	780K	Direct OA1000	96(DSDD) 622K
Discovery	96(DSDD)	624K	Eagle II/IV	96(SSDD) 386K
Eagle III	96(DSDD)	762K	Epic	96(DSDD) 782K
Facit DTC	96(SSDD)	308K	Heath H-37	96(SSDD) 308K
Heath H-37	96(DSDD)	624K	Honeywell	96(DSDD) 632K
ICL Model 25	96(DSDD)	628K	ICL Model 35	96(DSDD) 778K
IMS (TurboDos)	96(DSDD)	772K	Intel iPDS	96(DSDD) 608K
Ithaca 525	96(SSDD)	346K	Ithaca 525	96(DSDD) 628K
Lobo Max-80	96(DSDD)	698K	MACSYM 150/350	96(SSDD) 308K
Monroe OC8820	96(SSDD)	306K	Monroe 2000	96(DSDD) 628K
MultiTech MIC-504	96(DSDD)	698K	NCR Work Saver	96(DSDD) 544K
Olympia People	96(DSDD)	618K	OSM Zeus 4	96(DSDD) 620K
Otrona	96(DSDD)	770K	Philips P2000C	96(DSDD) 628K
Philips 3000	96(SSDD)	294K	Pied Piper	96(DSDD) 776K

Table
Disk Maker I Formats

(Continued on next page)

Sanyo 1200/50	96(DSDD) 620K	Sanyo 2000	96(SSDD) 302K
Sanyo 4050	96(DSDD) 620K	TeleVideo 1603	96(DSDD) 706K
Vector 4-S	96(DSDD) 712K	Vector VSX	96(DSDD) 710K
ALTOS	8"(SSDD) 446K	CCS (256b)	8"(SSDD) 482K
CCS (1024b)	8"(SSDD) 596K	Colonial Data SB80	8"(SSDD) 596K
Colonial Data SB80	8"(DSDD) 1208K	Columbia 1800	8"(SSDD) 592K
CompuPro (1024b)	8"(SSDD) 596K	CompuPro (1024b)	8"(DSDD) 1192K
CP/M SSSD Standard	8"(SSSD) 241K	Delta Products 125	8"(DSDD) 1212K
Harris 1685-50MFT	8"(SSDD) 596K	Harris 1685-50MFT	8"(DSDD) 968K
Harris 1685-50MFT	8"(DSDD) 1196K	IBM 3740 TO Direct	8"(SSSD) 248K
IBM 5520 WP	8"(DSDD) 1132K	IBM 128b no skew	8"(SSSD) 241K
IBM 256b no skew	8"(SSDD) 482K	Insight IQ-120	8"(SSDD) 484K
Megaflex Apple	8"(SSDD) 558K	Molecular	8"(SSDD) 496K
NEC APC CP/M-86	8"(DSDD) 980K	S. D. Sales (128b)	8"(SSDD) 464K
S. D. Sales (256b)	8"(SSDD) 476K	Tarbell (128b)	8"(SSDD) 472K
TRS-80 II,12/6 P&T	8"(SSDD) 596K	TRS-80 II,12/6 P&T	8"(DSDD) 1210K
TRS-80 Lifeboat	8"(SSDD) 482K	TRS-80 Lifeboat	8"(SSDD) 596K
Vector 2800	8"(DSDD) 984K	Zenith Z-100	8"(SSDD) 482K
Zenith Z-100	8"(DSDD) 980K	ACT Apricot MSDOS	3"(SSDD) 314K
HP-150	MSDOS 3"(SSDD) 258K	Intertec H-S MSDOS	3"(SSDD) 395K
IBM PC	PCDOS1.1(SSDD) 156K	IBM PC	PCDOS1.1(DSDD) 315K
IBM PC	PCDOS2.0(SSDD) 175K	IBM PC	PCDOS2.0(DSDD) 354K
IBM PC-AT	PCDOS3.0(DSDD) 1186K	Kaypro 10	MSDOS (DSDD) 354K
Wang	MSDOS (DSDD) 354K	Zenith	ZDOS 1.1(DSDD) 315K
BurroughsB25	MSDOS 96(DSDD) 610K	DEC Rainbow	MSDOS 96(SSDD) 384K
Eagle 1600	MSDOS 96(DSDD) 785K	Monroe	MSDOS 96(DSDD) 712K
NCR D-M "M"	MSDOS 96(DSDD) 790K	Otrona	MSDOS 96(DSDD) 712K
Sanyo MBC	MSDOS 96(DSDD) 792K	Tandy 2000	MSDOS 96(DSDD) 714K
Lomas Data	MSDOS 8"(SSSD) 239K	NEC APC	MSDOS 8"(DSDD) 1221K
Standard	MSDOS 8"(SSSD) 246K	Zenith V1.1	ZDOS 8"(SSSD) 245K

NOTE: all formats 5 1/4" 48 tpi, CP/M-80/86 unless noted as:

3" - 3 1/2 inch format

8" - 8 inch format

96 - 96 tpi 5 1/4 inch format

SS = Single Sided; DS = Double Sided

SD = Single Density; DD = Double Density

Table
Disk Maker I Formats

8-inch formats and the IBM PCAT drive require at least a 4 MHz system clock; otherwise, a 2 MHz clock should be sufficient.

Installation of the Disk Maker I hardware is as easy as plugging the disk controller board into an empty slot in your S-100 bus motherboard and connecting the disk cable between the disk cabinet and the controller board. A menu-driven installation program allows you to customize some features of the system to your hardware and personal taste; the installation process takes only a few minutes.

How It Works

Disk Maker I is simple to use. To format a disk, type DMFORM to invoke the format program, then choose a format by number from the on-screen menu. You can also invoke DMFORM by making a format choice on the command line; the program proceeds without showing you the menu. You can return to the menu if you want to change your selection. DMFORM will verify a disk after formatting if you set a permanent option with DMINSTALL or specify verification on the command line when you invoke DMFORM.

The DMSET program gives you access to the Disk Maker disk drives. You use DMSET the same way you use DMFORM: you can choose from a menu or go directly to work by specifying a format on the command line. You can set each Disk Maker drive to a different format. Every time you warm boot, the Disk Maker software displays the disk formats you have selected.

DMSET installs a special BIOS under your normal CP/M system; you can run any CP/M program that will fit in the somewhat reduced memory left for applications. Even though

there is less memory, I have found it adequate for running any of my utilities, dBASE II or my text editor (which does bidirectional scrolling through a disk file). You probably will notice the memory reduction the most when you run a spreadsheet program or any other application that uses all available memory but does not scroll data to and from disk, the way many text editors do.

Because of the radically different structure of MSDOS files, you need a special program called TOMS (TO MSDOS) to access individual disk files on a MSDOS disk. You cannot run another program while using TOMS; if you want a CP/M application to access a MSDOS data file, you first must transfer the file to a CP/M format. TOMS is like a small part of the MSDOS operating system, providing the commands DIR, COPY, TYPE, and ERASE. TOMS cannot deal with subdirectories, so all files must be in the root directory on a MSDOS disk coming into the system; all files written by TOMS are in the root directory.

Performance

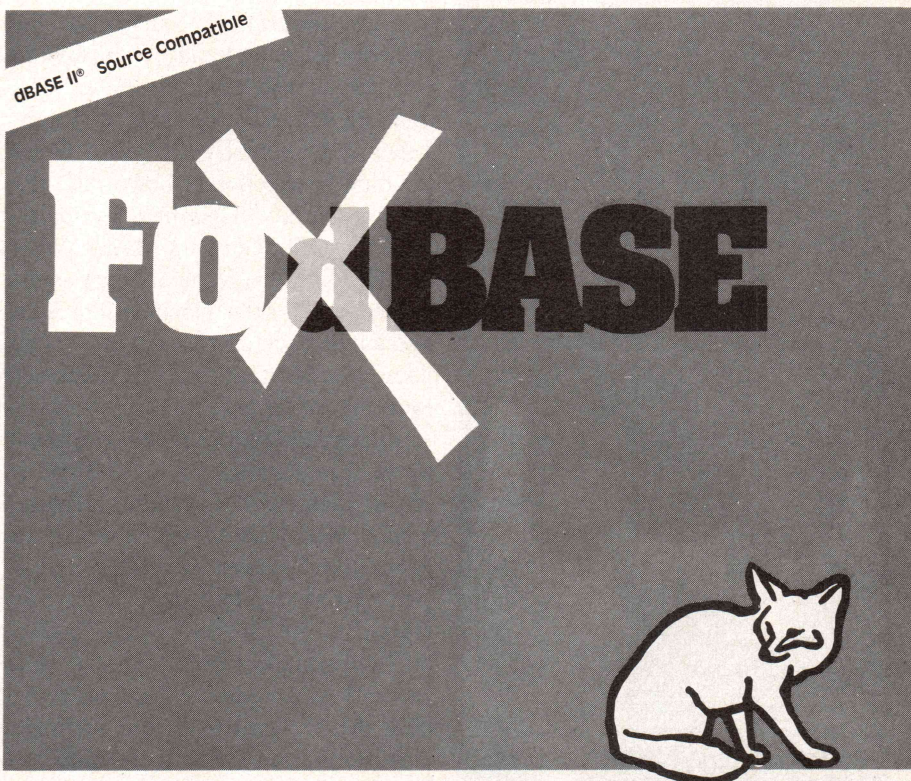
I have been using Disk Maker I for well over a year. The capabilities of the system are impressive, but even more impressive is the support New Generation Systems provides. I was an early user and, as you might expect, everything did not start out working perfectly. One annoying problem was caused by the routing of the data cable in the disk drive cabinet. The cable was picking up noise from the power supply, which gave me errors when reading the inner tracks of the 48 tpi drive. I solved the problem by changing the routing of the cable at the suggestion of New Generation Systems, but in the meantime, I received a replacement 48 tpi drive that had been specially aligned and certified. I have always received courteous and competent assistance anytime I have had a reason to call New Generation Systems with a problem or a question. You can expect the same treatment. (I was dealing with New Generation Systems as just another user, before I had thought of doing this review.)

The Disk Maker I hardware con-

sists of off-the-shelf components integrated by New Generation Systems—nothing “razzle-dazzle” but obviously chosen carefully for quality and value. What makes the Disk Maker I package outstanding is the software and support. The programs perform well and are easy to use. The no-nonsense operator interface was designed for users who need to get things done quickly.

Although the software is not friendly in the current sense of having elaborate menus and on-line help fa-

cilities, each user prompt and program response is clear in its meaning. With one exception, which I will discuss later, all errors are trapped cleanly and appropriate error messages appear on the screen. The user interface is appropriate for the intended users of this package: software professionals, experienced computer users, and serious hobbyists with a need to transfer files between diverse disk formats. Operation is simple enough for a novice to use the system, too.



Evolution.

Now FoxBASE, the dBASE II source-compatible interpreter/compiler, is even better than before. Automatic 8087 co-processor support allows you to perform numeric computations with lightning speed. Fourteen-digit precision gives you 40% greater accuracy than dBASE II. The fact that FoxBASE is not copy protected means you can easily load it onto your hard disk. What's more, FoxBASE comes complete with a NO-RISK demo plan.

Of course, FoxBASE still offers all of the features that dBASE II does . . . PLUS

- Runs 3 to 20 times faster • Permits up to 48 fields/record...50% more than dBASE II
- Supports full type-ahead • Compiles pro-

gram sources into compact object code • Has twice as many variables • Comes with a sophisticated online manual and HELP facility.

FoxBASE is currently available on a wide range of machines: IBM-PC, IBM-PC/XT/AT, COMPAQ & IBM compatibles, TI Professional, DG Desktop, and DG MV-Series to name just a few. And it will soon be available on the Molecular and NCR Tower computers as well. Call or write today for more information.

MS-DOS:	Development Pkg.	\$395
	Runtime Pkg.	\$695
AOS/VS:	Development Pkg.	\$995
	Runtime Pkg.	\$1995

UNIX and XENIX: (To Be Announced)

Developed by

DACOR

COMPUTER SYSTEMS

dBASE II is a trademark of Ashton-Tate.
UNIX is a trademark of AT & T.
FoxBASE is a trademark of Fox Software Inc.

FoxBASE™
from FOX SOFTWARE INC.

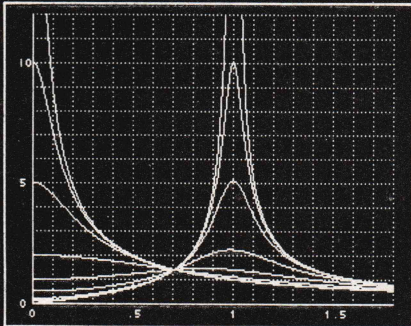


13330 Bishop Road, P.O. Box 269, Bowling Green, OH 43402 / 419-354-3981 / TWX 810-499-2989

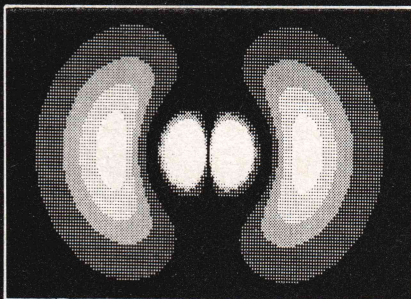
isys FORTH

for the Apple®] [

Fixed point speed can rival that of floating point hardware. But the details have been a well kept secret—until now. The following graphs were generated by fixed point examples from the ISYS FORTH manual.



Parallel Resonance with Damping
BASIC 213 sec ISYS FORTH 27 sec



Hydrogen 3p Orbital Cross-section
BASIC 492 sec ISYS FORTH 39 sec

- **Fast native code compilation.** Sieve benchmark: 33 sec
- **Floating Point**—single precision with transcendental
- **Graphics**—turtle & cartesian with 70-column character set
- **Double Precision** including D*/
- **DOS 3.3** Files read & written
- **FORTH-83** with standard blocks
- **Full-Screen Editor**
- **Formatter** for word processing
- **Macro Assembler**
- **Price:** \$99, no extra charges

ILLYES SYSTEMS

PO Box 2516, Sta A
Champaign, IL 61820

Technical Information:
217/359-6039, mornings

For any Apple] [model, 48K or larger.
Apple is a registered trademark of Apple Computer.

The Disk Maker I (and Disk Maker II) documentation is well organized and appropriate for the intended users. The *User's Manual* for Disk Maker I contains over 50 pages with a good table of contents in a three-ring binder. The table of contents is adequate for using the manual: no index is provided and none is needed. The manual, prepared with a word processor, is much cleaner in appearance than most manuals prepared this way. It is relatively easy to flip through the pages and spot whatever you are looking for.

New Generation Systems has assumed that you already know how to use CP/M and your computer. The manual will say "copy this file to another disk" rather than give a blow-by-blow description of how to use the file copy utility. There is a section titled "How To Run Disk Maker Without Reading The Manual" for the impatient and for the technical-minded a section on each program called "How xxxx Works" that explains the inner workings of the program.

I have validated Disk Maker's ability to format, read, or write disks in about 20 of the 170 available formats. I have used Disk Maker primarily to move files between an S-100 system and an Osborne I (double density) or between the S-100 system and a PCDOS system. Other than the read errors I encountered (now fixed), the Disk Maker hardware and software have performed flawlessly for me.

I have encountered a few instances where it has been necessary to bulk erase a disk and format it with Disk Maker I because, if I wrote on a disk already formatted by the receiving system, that system could not read the disk.

Although Disk Maker I is easy to run, one enhancement would be welcome. The Disk Maker I software assumes you know what disk format you are dealing with. If you tell it you are feeding it a disk of format X and then give it a disk of format Y, you may or may not get an error message; the program you run could go merrily along without complaint, producing either garbage files or a disk that the target system cannot read. The Disk Maker software cannot identify posi-

tively that you have inserted the correct disk format in the drive being written to because no standard means of identification exists for a blank formatted disk; even if you determine the sector size, sectors per track, and so forth, you still cannot deduce the directory size and other characteristics needed for positive identification.

WHATDISK enables you to determine which of several similar formats a disk might be, if you already know that the disk is for IBM PCDOS, Heath/Zenith Z-100, or Morrow Micro Decision. A generalized WHATDISK program to identify any disk format would be very useful. Such a program is impossible to write, however, for the above reasons.

What You Pay

The basic Disk Maker I, which includes the controller board, one 48 tpi, DSDD 5¼-inch drive, and the Disk Maker software is \$1695. The 96 tpi, DSDD drive is an additional \$395. The 8-inch drive and cabinet for Disk Maker I is \$849. The 3½-inch drive costs \$295, and the PCAT drive is \$495. The word processing transfer utilities package is \$295, and the disk testing software is \$150 plus \$40 each for the 48 and 96 tpi Dysan test disks. You also can purchase the Disk Maker I controller and software with no drives.

The basic Disk Maker II system is \$3395 for the stand-alone 6 MHz Z80B system with one DSDD 8-inch drive, one 48 tpi, DSDD 5¼-inch drive, and CP/M 2.2; you only need to add a terminal. The option prices are the same as for Disk Maker I, and a second DSDD 8-inch drive costs \$600.

Are these prices reasonable? It depends on what you need. Programs for the Kaypro, IBM PC, Morrow, and other computers that allow you to read, write, and format a number of 5¼-inch formats cost less than \$100, but don't ask them to recognize an 8-inch disk drive, a PCAT drive, or a 3½-inch drive. At the other end of the spectrum are systems that claim less capability than Disk Maker II and cost over \$5000. Disk Maker I and Disk Maker II offer outstanding versatility and performance for the money.

NEW!

RELOCATABLE Z-80 MACRO ASSEMBLER FROM MITEK

It's a real bargain! Here's why:

- Only \$49.95 plus shipping
- 8080 to Z-80 Source Code Converter
- Generates Microsoft compatible REL files or INTEL compatible hex files
- Compatible with Digital Research macro assemblers MAC & RMAC
- Generates Digital Research compatible SYM files
- Full Zilog mnemonics
- INCLUDE and MACLIB files
- Conditional assembly
- Separate data, program, common and absolute program spaces
- Customize the Macro Assembler to your requirements with installation program
- Cross-reference Generation
- Z-80 Linker and Library Manager for Microsoft compatible REL files available as a total package with Macro Assembler for only \$95.00 plus shipping
- Manual only is \$15

TO ORDER, CALL TOLL FREE: 1-800-367-5134, ext. 804
For information or technical assistance: 1-808-623-6361

Specify desired 5 1/4" or 8" soft-sectored format. Personal check, cashier's check, money order, VISA, MC, or COD welcomed. Include \$5 for postage and handling.

MITEK P.O. Box 2151
Honolulu, HI 96805

Z-80 is a trademark of Zilog, Inc. MAC, RMAC, and ZSID are trademarks of Digital Research, Inc.

Circle no. 66 on reader service card.

INSIGHT™

EXPERT SYSTEMS

"INSIGHT is essentially the equivalent or better than any other tool available for the personal computer."

Paul Harmon, author of Expert Systems, Artificial Intelligence in Business

Turn your PC into an expert.

Give it Insight, or give it Insight 2. Both let you create knowledge base systems using any PC-compatible text editor.

Insight not only simplifies access to lots of information, it analyzes and offers solutions. For entry-level operators it's a perfect procedural training package to help build and implement knowledge base software.

Insight 2 is more than just an "expert." It's a knowledge base engineering tool with application capabilities. It can call up Pascal programs, read and write dBASE II® files, and its decision-making process can tie in directly to your existing databases. Run-only versions also can be developed and distributed.

Two unique packages from the same expert idea.

Insight™ (\$95) and Insight 2™ (\$485) run on the IBM® PC, DEC® Rainbow, and Victor® 9000.



**Level
Five
Research, Inc.**

4980 South A-1-A

Melbourne Beach, Florida 32951

(305) 729-9046

Circle no. 53 on reader service card.

WALTZ LISP

The universal, super-efficient
Lisp for PC-DOS, MS-DOS,
CP/M-86 and CP/M-80
systems.

Waltz Lisp is a very powerful and complete implementation of Lisp. It is similar to Franz (the Lisp running under Unix), and is substantially compatible with MacLisp and other mainframe Lisps.

Ultra fast. In independent tests, **Waltz Lisp** was up to twenty(!) times faster than competing microcomputer Lisps.

Easy to use. The interpreter can directly load program files created with any ASCII text editor. Full debugging and error handling facilities are available at all times. No debuggers to link or load.

Practical. Random file access, binary file support, and extensive string operations make **Waltz Lisp** suitable for general programming. A text-file difference program and other utilities are included in the package.

Full Lisp. Functions of type *lambda (expr), nlambda (expr), lexp, macro*. Splicing and non-splicing character macros. Full suite of mappers, iterators, etc. Long integers (up to 611 digits). Fast list sorting using user defined comparison predicates. Built-in prettyprinting and formatting facilities. Over 250 functions in all.

Flexible. Transparent (yet programmable) handling of undefined function references allows large programs to reside partially on disk at run time. Optional automatic loading of initialization file. User control over all aspects of the system. Assembly language interface.

Superbly documented. Each function is described in detail. The 300+ page manual includes an exhaustive index and hundreds of illustrative examples.

Order **Waltz Lisp** now and receive **free** our **PROLOG** interpreter

Clog Prolog is a tiny (but very complete) Prolog implementation written entirely in **Waltz Lisp**. In addition to the full source code, the package includes a 50 page **Clog** manual.

16-bit versions require DOS 2.x or CP/M-86 and 90K RAM (more recommended). Z-80 version requires CP/M 2.x or 3.x and 48K RAM minimum. **Waltz Lisp** runs on hundreds of different computer models and is available in all disk formats.



\$169*

*Manual only: \$30 (refundable with order). Foreign orders: add \$5 for surface mail, \$20 for airmail. COD add \$3. Apple CP/M, hard sector, and 3" formats add \$15. MC/Visa accepted.

For further information or to order call
1-800-LIP-4000 DEPT. 21

In Oregon and outside USA call 1-503-684-3000

PC™
PRO CODE
INTERNATIONAL

15930 SW Colony Pl.,
Portland, OR 97224

Circle no. 73 on reader service card.

Conclusion

Disk Maker I is a solid product backed by a competent and cooperative company that operates as if it wants to remain in business for a long time. If you are a software developer, software distributor, user group, large office, or serious hobbyist with the need to exchange files between computers with various disk formats, you should seriously consider Disk Maker I or Disk Maker II as an answer to your needs.

AMPRO Little Board and Bookshelf Computers

Company: AMPRO Computers, Inc., 67 East Evelyn Ave., Mountain View, CA 94041 (415) 962-0230.

Price: \$349

Circle Reader Service No. 105

Reviewed by Richard Conn

The heart of the AMPRO Bookshelf Computer® is the AMPRO Little Board®, a full-featured single-board computer that measures only 5¼-inch by 7¼-inch and is designed for mounting on the back of a standard 5¼-inch minifloppy disk drive. Although it is small in size, the AMPRO Little Board is big in features: the Little Board contains all of the hardware necessary to support a conventional CP/M environment.

The Little Board is based around a 4 MHz Z80A microprocessor and the associated Zilog Z80 family of support chips. On board this small computer is:

- A 4 MHz Z80A microprocessor
- 64K of dynamic RAM
- One 4K 2732-type EPROM
- A Z80A CTC (Counter/Timer Circuit)
- A Z80 DART (Dual Asynchronous Receiver/Transmitter)
- A parallel output port (based on D-type latches)
- A Western Digital 1770 Floppy Disk Controller Chip

The AMPRO Bookshelf Computer is an AMPRO Little Board computer housed in an attractive case with one or two 5¼-inch floppies, built-in

power supply, two 25-pin EIA RS-232C connectors, one Centronics connector, and a connector for tagging on up to two more 5¼-inch floppies. An ON/OFF switch and a RESET switch are available on the front panel.

The AMPRO Little Board Computer

Hardware

The Little Board has two possible memory configurations, which depend on the setting of an EPROM-enable bit in the Board Control Register. Enabling the EPROM establishes the following memory configuration:

Address (Hex)	Memory Element
0000 - 0FFF	2732 EPROM
1000 - 7FFF	2732 EPROM duplicated on each 4K group
8000 - FFFF	RAM

If the EPROM is disabled, all memory from 0 to 0FFFFH is enabled as RAM.

A Z80 DART and RS-232C line drivers provide the serial input/output ports. The DART is an asynchronous device, providing for baud rates up to 38,400 on its A channel and 9600 on its B channel in the Little Board configuration. The Little Board does not implement all of the RS-232C signals on its two serial channels. Only the following signals are provided:

- Serial Data Output (RS-232C pin 3)
- Serial Data Input (RS-232C pin 2)
- Request to Send Handshaking Output (RS-232C pin 5)
- Clear to Send Handshaking Input (RS-232C pin 20)

Both channels of the DART are wired as DCE (Data Communications Equipment), which is the complement of DTE (Data Terminal Equipment)—the wiring of the RS-232C connector on computer terminals. Notably missing from this list is the Data Carrier Detect signal. You may use the Clear to Send signal in place of Data Carrier Detect.

The parallel port is an output-only

port, wired in a Centronics-standard configuration. The Z80 CTC on the Little Board provides four counter/timers for use in conjunction with the hardware and software of the computer.

The Western Digital 1770 Floppy Disk Controller chip used in the Little Board design can work with just under 400K per disk on double-sided, 48 tracks-per-inch (TPI) drives and just under 800K per disk on double-sided, 96 TPI drives.

Software

The AMPRO Little Board is supplied with one copy of CP/M 2.2, which uses the ZCPR3 Command Processor in place of the CP/M CCP. Although only the ZCPR3 Command Processor, in a less than maximum configuration, is provided with the Little Board, you can obtain a more complete ZCPR3 implementation from Echelon (see below).

The ZCPR3 configuration distributed with the Little Board provides CP/M 2.2 compatibility while simultaneously presenting the following additional features:

- Multiple commands per line, separated by semicolons, enabling command lines like DIR;ERA *.BAK;DIR
- Automatic command search path that searches through directories in the following sequence when looking for COM files: (1) current disk, current user; (2) current disk, user 0; (3) disk A, current user; (4) disk A, user 0; (5) disk A, user 15; and (6) current disk, user 15
- Four-element shell stack, so shells other than the ZCPR3 Command Processor can act as the user interface to the system
- Several built-in commands, including: GET—load file anywhere in memory; JUMP—call subroutine anywhere in memory; and LIST—print file on printer
- Extended directory references over normal CP/M: D:—reference disk (e.g., TYPE B:MYFILE.TXT); U:—reference user area (e.g., DIR 5:); and DU:—reference disk and user area (e.g.,

See articles in *Dr. Dobb's, Computer Language, Byte*, and other such magazines for more information on ZCPR3. The authorized agent for ZCPR3 is Echelon, Inc., 101 First St., Los Altos, CA 94022 (415) 948-3820. The ZCPR3 RCP/M and BBS are also available to you at (415) 489-9005.

The nine standard CP/M utility programs (ED, DDT, PIP, etc.) and thirteen additional utility programs are supplied with the Little Board. Included in this list is MULTISK, which allows the user to dynamically select one drive to support any one of over 40 different 5¼-inch floppy disk formats, and 48TPI, which allows the user to read 48 TPI disks on a 96 TPI drive.

One nice option that AMPRO gives the customer is that of purchasing the source code to several key programs at a reasonable price. The customer may purchase the Technical Support Software package, which includes the source files for MULTISK, the BIOS, and the BOOT.

Pricing

The AMPRO Little Board computer costs \$349. This price includes the single-board computer and all of the software listed above. An extra \$50 buys the Technical Support Software package.

The AMPRO Bookshelf Computer

Hardware

As mentioned above, the AMPRO Bookshelf Computer consists of an AMPRO Little Board, one or two 5¼-inch minifloppy disk drives, a built-in power supply, two 25-pin EIA RS-232C connectors, one Centronics connector, one connector for attaching up to two external minifloppy disk drives, and ON/OFF and RESET switches on the front panel. The Bookshelf is available in the following models:

Model	Configuration
121	One double-sided, 48 TPI drive (400K)
122	Two double-sided, 48 TPI drives (800K)
141	One double-sided,

96 TPI drive (800K)
142 Two double-sided,
96 TPI drives (1600K)

Software

All of the software provided with the AMPRO Little Board is also provided with the AMPRO Bookshelf Computer. In addition, a ZCPR3 shell designed specifically for use on the AMPRO computer is provided (in object code only). FRIENDLY is a screen-oriented tool that provides a directory of files to the user of the current disk and allows him or her to perform a variety of operations on the files displayed. By simply moving a pointer to a desired file and pressing a few keys, a FRIENDLY user can copy a file, delete a file, view a file on the console, print a file on the printer, run a program, mark a file for later delete or copy with a group, perform a sequence of commands on the file based upon a preprogrammed menu, and perform many more operations.

FRIENDLY is easy to use, usually requiring only a few minutes to learn, and it removes casual computer users from the CP/M command environment, relieving them of the necessity of learning how to use CP/M in order to use the computer.

A second software package provided with the AMPRO Bookshelf is T/MAKER, a product of T/MAKER Company. T/MAKER provides a screen-oriented, memory-based editor, text formatter, electronic spreadsheet, bar graph generator, and file management system in one integrated package. T/MAKER will run under conventional CP/M, CP/M enhanced by the ZCPR3 Command Processor, or FRIENDLY. Several articles have already been published on T/MAKER, and you can obtain further information by writing to T/MAKER Company, P.O. Box 6430, Falls Church, VA 22046.

Documentation

The documentation provided with the AMPRO Little Board is the *Little Board User's Manual*, a mainly hardware manual that describes the Little Board in great detail. It is well written and clear if you have a background in digital electronics.

NEW FEATURES

(Free update for our early customers!)

- Edit & Load multiple memory resident files.
- Complete 8087 assembler mnemonics.
- High level 8087 support. Full range transcendentals (tan, sin, cos, arctan, logs and exponentials) Data type conversion and I/O formatting.
- High level interrupt support. Execute Forth words from within machine code primitives.
- 80186 Assembler extensions for Tandy 2000, etc.
- Video/Graphics interface for Data General Desktop Model 10

HS / FORTH

- Fully Optimized & Tested for:
IBM-PC IBM-XT IBM-JR
COMPAQ EAGLE-PC-2
TANDY 2000 CORONA
LEADING EDGE
(Identical version runs on almost all MSDOS compatibles!)
- Graphics & Text
(including windowed scrolling)
- Music - foreground and background
includes multi-tasking example
- Includes Forth-79 and Forth-83
- File and/or Screen interfaces
- Segment Management Support
- Full megabyte - programs or data
- Complete Assembler
(interactive, easy to use & learn)
- Compare
BYTE Sieve Benchmark jan 83
HS/FORTH 47 sec BASIC 2000 sec
w/AUTO-OPT 9 sec Assembler 5 sec
other Forths (mostly 64k) 70-140 sec

FASTEST FORTH SYSTEM
AVAILABLE.

TWICE AS FAST AS OTHER
FULL MEGABYTE FORTHS!

(TEN TIMES FASTER WHEN USING AUTO-OPT!)

HS/FORTH, complete system only: \$250.

VISA Mastercard

Add \$10. shipping and handling

HARVARD SOFTWARES

P.O. Box 2579
Springfield, OH 45501
513/390-2087

No documentation on CP/M or the standard CP/M programs is provided. The first pages of the manual, however, give pointers to where to obtain this documentation. Also, no documentation other than a brief overview of ZCPR3 is provided. Again, the manual gives pointers to Echelon.

If you buy the AMPRO Bookshelf Computer, you get the *Little Board User's Manual* as well as documentation on FRIENDLY and T/MAKER. The documentation on FRIENDLY and T/MAKER is oriented to computer users having some limited experience with CP/M.

General Impressions and Comments

I have used the AMPRO Bookshelf for several months now. I started with the Model 122, which had two 400K, 48 TPI drives, and am now using the Model 142, which has two 800K, 96 TPI drives. I find its clean design, speed, and reliability to be excellent. Given my background in CP/M and digital electronics, I found all of the documentation on the Little Board/Bookshelf to be readable and understandable.

The Bookshelf is a nice computer for any user, especially with the FRIENDLY shell as a front end. Although it is a good choice for the first-time computer user, as with any computer, you must overcome a learning curve before you can expect effective use of the computer.

The AMPRO BIOS performs well. The speed of the disks with this BIOS is nice: the disks are faster than most 5¼-inch disk systems I have observed and many 8-inch disk systems I have used. At 800K per disk with the Model 142 (96 TPI drives), I find the disk capacity also to be quite reasonable.

Overall, the AMPRO Bookshelf and AMPRO Little Board are good computers, but there are a few drawbacks for some applications:

- (1) Lack of documentation on CP/M and ZCPR3. However, AMPRO supplies pointers to where you can obtain such documentation, at additional cost.
- (2) Limitation of serial I/O ports to only two serial RS-232C ports and one parallel port. Nevertheless, if

your needs call for no more than a terminal, modem, and printer (as many popular computers are configured today), the AMPRO computers are adequate.

(3) Lack of hard disk support and expandability. The Little Board does not support hard disks or other such devices in its current form. You may buy, however, the Little Board SCSI/PLUS Adapter for \$99. This adapter "piggy-backs" on the Little Board and allows you to attach a Winchester disk drive, slave processor and device boards (perhaps for I/O), and other facilities. This board, with some effort, may eliminate drawbacks 2 and 3.

I am excited about the Little Board and the Bookshelf for their applications in several environments. As a stand-alone personal computer with limited I/O resources and disk space, these computers are good choices. The SCSI/PLUS Adapter promises to expand their capabilities.

For embedded computer applications in which 4K of code is enough to run the application program, the Little Board provides a nice, inexpensive self-contained computer. The 4K 2732 EPROM can be programmed for the application, and 32K of RAM is available for data. The two RS-232C ports are there, as well as the parallel output-only port. No disks would be needed for this type of application. For embedded computer applications with disks, the AMPRO Little Board would "piggy-back" on a 5¼-inch 96 TPI minifloppy and feed off of the floppy's power supply. This would provide a very small embedded computer with 800K of disk and 64K of RAM.

As a slave computer to another computer, the AMPRO Bookshelf with its two disks could act as a background batch processor, communications controller (for a BBS), printer spooler, or other such intelligent and flexible slave device.

Overall, I like the AMPRO Little Board and AMPRO Bookshelf and intend to continue using my Bookshelf for a variety of slave computer and stand-alone applications (such as computer assistance for talks and presentations). I recommend it with

some reservation to first-time computer users and with no reservation to embedded applications designers and more experienced computer users.

BetterBASIC, Version 1.1

Company: Summit Software,
P.O. Box 99, Babson Park,
Wellesley, MA 02157
(617) 235-0729

Computer: IBM PC or close compatible with minimum 192K and one disk drive

Price: BetterBASIC \$199; 8087 Math Module \$99; Runtime System \$250

Circle Reader Service No. 107

Reviewed by Matthew Trask

Among the reasons that Summit Software gives for purchasing BetterBASIC are full 640K memory support, separately compiled program modules, language extensibility, window support, 8087 support, and incremental compilation. Although these reasons would be persuasive in the case of another programming language, I question whether they will induce many programmers to shell out \$199, the cost of BetterBASIC, when some form of GW BASIC (aka BASICA) is provided at no charge with most IBM type computers these days and an excellent Pascal compiler can be purchased for less than fifty dollars. In order to help programmers make a decision on this purchase, I will discuss the unique features of this programming system, benchmark its performance, and examine the difficulties of converting programs from GW BASIC to BetterBASIC.

Installation

Although the package can be run right out of the box by using its default configuration, you will probably want to configure it to suit your hardware environment. B.COM is the executable portion of BetterBASIC and it uses the file B.CNF to determine which of the other modules to load for operation. Table 1 (page 110) is a listing of all the modules that are provided on the BetterBASIC disk and Table 2 (page 110) shows the default configuration. An ASCII editor can be used

Electronic Circuit Analysis

- New release
- Transient, AC, DC analysis
- Full nonlinear
- Over 200 nodes
- Full editing
- Macro circuits
- Worst case, Monte-Carlo
- Temperature effects
- Frequency dependent parts
- Time dependent parts

For MS-DOS. 192k minimum.
\$395.00

Tatum Labs
33 Main Street
Newtown, CT 06470
(203) 426-2184

Circle no. 82 on reader service card.

A Professional Quality Z80/8080/8085 Disassembler WHEN YOU NEED SOURCE FOR YOUR CODE you need REVAS 3

REVAS interactively helps you:

- Analyse your software for modification
- disassemble files as large as 64K
- Assign Real labels in the disassembly
- Insert COMMENTS in the disassembly
- Generate a Cross Reference (XREF) listing

A 60 page manual shows how the powerful **REVAS** command set gives you instant control over I/O to files, printer, or console; how to do a disassembly; and even how the disassembler works! You get on line help, your choice of assembler mnemonics, control of data interpretation, and calculation in any number base!

REVAS runs in Z80 CPM computers; is available on 8" SSSD (standard), RAINBOW, and other (ask) formats

Price: \$90.00 (plus applicable tax), Manual only: \$15.00

REVASCO
6032 Chariton Ave., Los Angeles, CA 90056
Voice: (213) 649-3575 Modem: (213) 670-9465

Circle no. 80 on reader service card.

Write it once!

MasterFORTH

Portable programming environment



Whether you program on the **Macintosh**, the **IBM PC**, an **Apple II** series, a **CP/M** system, or the **Commodore 64**, your program will run unchanged on all the rest. If you write for yourself, MasterFORTH will protect your investment. If you write for others, it will expand your marketplace.



MasterFORTH is a state-of-the-art implementation of the Forth computer language. Forth is interactive – you have immediate feedback as you program, every step of the way.

Forth is fast, too, and you can use its built-in macro assembler to make it even faster. MasterFORTH's relocatable utilities, transient definitions, and headerless code let you pack a lot more program into your memory. The resident debugger lets you decompile, breakpoint, and trace your way through most programming problems. A string package, file interface, and full screen editor are all standard features.

MasterFORTH exactly matches the Forth-83 Standard dialect described in *Mastering Forth* by Anderson and Tracy (Brady, 1984). The standard package includes the book and over 100 pages of supplementary documentation.

MasterFORTH standard package

Macintosh	\$125
IBM PC and PC Jr. (MS DOS 2.1)	125
Apple II, II+, IIe, IIc (DOS 3.3)	100
CP/M 2.X (in several formats)	100
Commodore 64	100

Extensions

Floating Point (1984 FVG standard)	\$40
Graphics (Apple II series)	40
Module relocater (with utility sources)	60
Printed source listing (each)	35

Publications

<i>Mastering Forth</i> (additional copies)	\$18
<i>Thinking Forth</i> by Leo Brodie	16
<i>Forth-83 International Standard</i>	15
<i>Rochester Bibliography</i> , 2nd ed.	15
<i>1984 Rochester Conference</i>	25
<i>1984 J1 of Forth Appl. & Res.</i> 2(2)	15
<i>1983 FORML Conference</i>	25



MICROMOTION

12077 Wilshire Blvd., #506
Los Angeles, CA 90025
(213) 821-4340

Circle no. 132 on reader service card.

to add or delete modules and create new configurations. For example, users that don't have a graphics card may wish to delete GRAPHICS.IBM in order to free up memory for program use. I replaced FILE.DOS with FILE2.DOS in order to take advantage of DOS 2.x subdirectories and added SYSCALL.IBM in order to use the SHELL command and BIOS/DOS calls. You may specify the use of alternative configurations when starting the program by typing B-MYCON-

FIG/c where MYCONFIG is the name of the optional configuration file.

A Tandy 2000 version of the program is also available. The modules with the extension .IBM are all replaced with equivalent modules that end with .TDY.

Startup and Operation

After a sign-on message and copy-right notice you will see the first major difference between BetterBASIC and GW BASIC: instead of the usual

"60891 Bytes free" message, a 512K machine will display "Left:260304 bytes." All available system memory is used as BetterBASIC's workspace. This allows much larger programs to be written. BetterBASIC supports most of GW BASIC's commands, including the on-screen editing keys that are used on the IBM PC. Table 3 (page 111) contains a complete listing of GW BASIC commands that are different or not supported and a list of new commands that are found in BetterBASIC.

Every statement that you enter is checked for syntax and then compiled. If you make a mistake when entering a program line, an error message will be given immediately. For example:

```
40 PRINT (1 + 2 * 3
----- ^
") Expected
```

If the statement is entered correctly, it is compiled to virtual-machine code rather than being interpreted at run-time, thus giving a large speed improvement over the GW BASIC interpreter.

As in Pascal and C, all variables are declared at the beginning of the program. The exception to this is the AUTODEF command, which allows automatic declaration of a variable by the first assignment of a value. AUTODEF defaults to ON, but if you turn it off, the compiler will catch misspelled variable names. The data types that are supported in BetterBASIC are byte, integer, real, string (up to 32,767 characters), array, pointer, and structure. Structures are identical to Pascal's record data type and may contain any other data type. Arrays can contain byte, integer, real, and string data as well as structures and other arrays, thus allowing arrays of arrays, arrays of structures, and structures of arrays.

A BetterBASIC programming session superficially resembles one in GW BASIC in that lines are numbered and most of the command keywords are identical. When saving to disk, however, you must specify the file's extension because .BAS is not assumed by BetterBASIC. At the

B	COM	35312	1-03-85	1:00a
B	DEF	27020	1-03-85	1:00a
B	CNF	142	1-03-85	1:00a
MATH	BCD	8112	1-03-85	1:00a
CONSOLE	IBM	21392	1-03-85	1:00a
MAIN		3840	1-03-85	1:00a
FILE	DOS	19056	1-03-85	1:00a
GRAPHICS	IBM	10512	1-03-85	1:00a
PLAY	IBM	3136	1-03-85	1:00a
EVENT	IBM	7744	1-03-85	1:00a
FILE2	DOS	21072	1-03-85	1:00a
CHAIN	MOD	2704	1-03-85	1:00a
SYSCALL	IBM	3520	1-03-85	1:00a
COMM	BAS	8484	1-03-85	1:00a
PLIST	BAS	2542	1-03-85	1:00a
SECTOR	BAS	6702	1-03-85	1:00a
STRUC	BAS	2980	1-03-85	1:00a
SHELL	CNF	175	1-03-85	1:00a
CHAIN	SIZ	11	1-03-85	1:00a
READ	ME	126734	1-03-85	1:00a

20 File(s) 1024 bytes free

Table 1

Directory listing of the BetterBASIC diskette showing the many modules that make up the Better BASIC system.

```
MODULES=MATH.BCD
MODULES=CONSOLE.IBM
MODULES=MAIN
MODULES=FILE.DOS
MODULES=GRAPHICS.IBM
MODULES=EVENT.IBM
MODULES=PLAY.IBM
STATUS=ON
```

Table 2

The default B.CNF configuration file.

end of a session you can exit to DOS with the SYSTEM command or by typing BYE.

Unique Features of BetterBASIC

The best feature of BetterBASIC is the use of true procedures with variables of local scope that exist only inside the procedure. Global variables can be used by declaring them as EXTERNAL in the procedure and providing the actual declaration at a level outside the procedure. A procedure can be declared as RECURSIVE if its name is declared as EXTERNAL inside the procedure. GOSUB and RETURN are both supported, but as you

gain experience with BetterBASIC, they will probably fall by the wayside in favor of these Pascal-like, parameterized procedures. In addition to passing arguments by value or reference, you may specify optional and/or default arguments.

One particularly distinctive aspect of BetterBASIC is the ANY ARG declaration. This allows a procedure or function to determine what type of data was passed to it at runtime with the following functions:

TYPE() extracts the type of datum

DIM() extracts the dimensions of

a datum

SIZE() extracts the size of a datum

For example, A=TYPE(FOO) will assign the following values to A:

0	if FOO is a byte
1	if FOO is an integer
2	if FOO is a real
3	if FOO is a string
4	if FOO is a structure
10 + N	if FOO is an array of above type N

If TYPE(A) returns 3 for string you

GW BASIC Statements Not Supported in BetterBASIC

BLOAD	BSAVE	CDBL	CSNG	CVD
CVI CVS	CONT	DEFDBL	DEFINT	
DEFSTR	DEF FN	DEF USR	EQV	ERASE
ERL FIELD	FRE	GET	IMP	
MERGE	MKD	MKI	MKS	MOTOR
ON PEN	ON STRIG	PEN	PMAP	PUT
RESUME	STICK	STRIG	TROFF	TRON
USR VARPTR	VIEW	WEND	WINDOW	

New BetterBASIC Statements

ANY ARG	ASH	AUTODEF	BYE
BYTE	BYTE ARG	BYTE ARRAY	BYTE ARRAY STRUC
BYTE PTR	CHECK	CODE	CODE?
COLOR BORDER	COMPRESS	CONSTANT	DEFINE WINDOW
DEL\$	DO . . END DO	DO IF	DO UNTIL
DYNAMIC	EDIT proc	ERROR	EXIT
EXIT n LEVELS	EXTERNAL or EXT	FRAME WINDOW	HEADER
INPUT FROM	INS\$	INTEGER or INT	INT ARG
INT ARRAY	INT ARRAY STRUC	INT FUNCTION	INT PTR
INTERRUPT	INTERRUPT PROC	KEYWORD ARG	LINES\$
LIST ALL	LIST ARGS	LIST PROCS	LOWERS\$
MAIN	MAKE MODULE	MAKE PROGRAM	OFFSET
ON INTERRUPT	PRECISION	PROCEDURE	PUBLIC
		or PROC	
READCHR	READCHR FROM	READLINE	READLINE FROM
REAL	REAL ARRAY	REAL ARRAY	REAL ARG
		STRUC	
REAL FUNCTION	REAL PTR	REPEAT	REPEAT IF
RESTORE SCREEN	RESULT =	ROT	SAVE MODULE
SAVE SCREEN	SEG	SET	SH
SIZE	SPAN	STRING or STR	STR ARG
STR ARRAY	STR ARRAY STRUC	STR FUNCTION	STR PTR
STRUCTURE	UPPER\$	WHILE . . DO	XREF

Table 3
A comparison of BetterBASIC and GW BASIC Keywords

HIPPO-C™

The C compilers for the
Macintosh™

LEVEL 1

Powerful, expandable, yet affordable:

- A friendly, integrated environment complete with a screen editor, full K&R C compiler, linker, source-level debugger, tutorial, standard C library, and structure definition files.
- Access to over 400 Toolbox routines.
- Convenient access to serial ports and sound channels.
- Over 200 pages of documentation and many sample programs.
- Upgradable to Hippo-C Level 2 for \$250.00.
- New version 1.2.
- \$149.95

LEVEL 2

Professional C development system:

- Allows for the creation of large, stand-alone commercial applications.
- Comes with a screen editor, optimizing K&R C compiler, 68000 assembler, linker, C library, stdio package, full floating-point support (including math and trig functions), and structure definition files.
- "Glue" routines which allow easy access to Macintosh features.
- UNIX™-like shell which includes many powerful commands and utilities.
- Convenient access to over 500 Toolbox routines.
- Documentation, many sample programs, and sources.
- No royalties or license fees.
- You may obtain a non-copy protected disk by signing and returning a form along with \$25.00 to Hippopotamus.
- \$399.95

HIPPO-LOCK™

The Hippo Business Data
Security System

- Encrypts all of your important Macintosh files to protect them from curious eyes. Encrypts MacWrite documents, MacPaint pictures, data bases, spreadsheets, data files, application programs, etc.
- Useful for protecting confidential information such as business plans, payroll information, sales projections, employee reviews, management proposals, grades, etc....
- Choose between three levels of security.
- Uses the NBS DES Data Encryption Standard, used widely by major corporations and the U.S. Government.
- \$119.95

HIPPOTAMUS

SOFTWARE, INC.

1250 Oakmead Parkway Suite 210
Sunnyvale, CA 94086

(408) 738-1200

Dealer inquiries welcome. We accept credit cards, checks, and money orders. California residents add local sales tax. Please include \$10 for shipping and handling. Export of Hippo-Lock may be limited by law and subject to license. Macintosh is a trademark of Apple Computer, Inc. UNIX is a trademark of AT&T Bell Labs. Hippo-C and Hippo-Lock are trademarks of Hippopotamus Software, Inc. Please allow 1-2 weeks for delivery. Price, availability, and specifications subject to change without notice.

can determine its allocated size (not length) with SIZE(A). If FOO is a structure, DIM(A) will return the number of fields and TYPE(FOO(N)) will return the type of field N. This flexibility of optional/ambiguous parameter passing can be a tremendous improvement over Pascal's rigid requirements.

BetterBASIC has a concept called Procedure Families in which more than one procedure or function can have the same name with the only distinction being the parameter list. This allows BetterBASIC to select which one to use based on the type of argument that is used. As an example, the two procedures that follow can be used to give informative error messages on integer input, rather than GW BASIC's cryptic "Redo from start."

```
PROCEDURE:IntegerInput
INT ARG Value
10 INPUT Value
20 PRINT "Your integer is ",
    Value
```

```
PROCEDURE:IntegerInput.a
STRING ARG SomeString
10 PRINT SomeString, "is not a
    valid integer, please try again"
```

In this example, IntegerInput.a will be used any time a valid integer is not entered when IntegerInput is called. This technique can also be used to assign additional functionality to BetterBASIC's reserved words.

Functions are similar to procedures in that they share all the same data types. The difference is that they are used in expressions to return a value that is used in the expression. The last statement of a function must be RESULT= and the value that is to be returned.

Both procedures and functions can use KEYWORD ARGuments—a concept that is similar to Pascal's user-enumerated data types. If an argument is declared as

```
KEYWORD ARG:Color
    \RED\GREEN\BLUE
```

then the procedure will accept either RED, GREEN, or BLUE as valid data

for Color.

BetterBASIC has a full complement of flow-controlling block structures. In addition to the usual FOR...NEXT, IF...THEN, and WHILE...REPEAT, BetterBASIC has DO UNTIL...REPEAT, DO...REPEAT, DO n TIMES...REPEAT, DO...END DO, DO...REPEAT, DO IF...REPEAT IF, and more. GOTOs are implemented in a way that could placate even Dijkstra—they cannot be used to enter or exit a control block. The only way out of a control structure is via the EXIT statement, and then only to the statement that immediately follows the block.

Variables can be declared as absolute by adding a segment and offset after the declaration. BYTE ARRAY-(4000):VideoBuffer [&hB000:0] is an absolute reference to the IBM's 2000 character monochrome video refresh buffer. This technique can be used to access the BIOS communication area in low memory for equipment determination.

The BetterBASIC language can be extended by adding your own procedures and functions to the language as keywords. The MAKE MODULE command will convert all procedures and functions currently in memory to a module that can be added to the B.CNF configuration file. Like C language function libraries, useful procedures can be distributed to other BetterBASIC users as modules with a good degree of source code protection because the source code cannot be derived from a module. Summit Software is currently selling an 8087 module and expects to offer an interface module for the SoftCraft Btrieve system sometime this spring.

Programs can be prepared for standalone execution with the MAKE PROGRAM command and the optional Runtime System. The Runtime System consists of RUN.COM, RUN.***, and MKEXE.COM. RUN.COM will execute any program that has been prepared with MAKE PROGRAM and MKEXE.COM appends the RUN.*** runtime library to a program file for stand-alone execution.

Assembly language support is pretty straightforward with a very good example given in Appendix G of

the manual. The keyword CODE is used instead of BLOAD in order to load an assembler module into memory at runtime. CODE? can be used to report the absolute load address if debugging is necessary. The assembly language procedure is then invoked by CALL procedurename(arglist) just as in GW BASIC. An assembly language module can contain one or more procedures with a Symbol Table at the beginning that defines the entry point for each procedure in the module. The excerpt from the BetterBASIC manual in the Listing (page 116) will give the flavor of assembly language interfacing.

Also worthy of note are the XREF command and the debug window. XREF generates a symbol cross-reference of all user-declared identifiers and the line numbers on which they appear. Advanced programmers may wish to use the DOS debug program to test programs that are developed in BetterBASIC. If they start with DEBUG B and then execute BetterBASIC inside DEBUG, the [Ctrl-PgUp] keys will exit to the debug prompt.

Current Version

The current version of BetterBASIC is 1.1. I received the upgrade from 1.0 to 1.1 during the review. New features supported include CHAINing and CALLing modules as overlays, communication between overlays via disk, SYSTEM calls for access to BIOS and DOS functions, and a SHELL command that can be used to execute programs (such as COMMAND.COM) from within BetterBASIC. The new XMEM keyword allows data structures such as arrays up to the limits of system memory, not just 64K as in v1.0. Also, the release notes describe bug fixes that were applied to this new version.

Summit Software informs me that all registered users will be provided with free updates to v1.1, with a copy of the new documentation on the disk and an option to purchase printed documentation for \$15. I also received test versions of the MATH1.BNY and MATH2.BNY (single and double precision binary math) that are significantly faster than MATH.BCD, but less precise. By the

time you read this, both modules should be included in the standard BetterBASIC system in addition to the BCD math module.

Benchmarks

I compared the performance of BetterBASIC and GW BASIC by running Rich Malloy's four BASIC benchmark programs that I downloaded from the *Byte* bulletin board. These tests separate I/O from computation to give fairly accurate representations of speed. With one exception (see Table 4, page 114), BetterBASIC was the better performer by far. The exception was in the floating point calculations, where the BCD math module was about 30% slower than GW BASIC. The new single precision binary math module ran about twice as fast.

Round-off error can be of concern in business and financial programming. Although MATH1.BNY showed no error in this benchmark (compared to GW BASIC's -1.788 E-07 error), it is probably well worth the slower speed of MATH.BCD to have guaranteed zero error in financial calculations.

The Runtime System, unlike the BASCOM compiler, will not show an improvement in execution speed. This is because BASCOM generates native 8086 code while MKEXE and RUNCOM only provide a stand-alone BetterBASIC environment to run the virtual-machine code that is produced by the MAKE PROGRAM command.

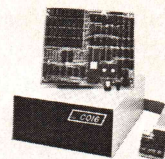
These benchmarks were run on an IBM PC with 512K of RAM, 10Mb hard disk, and a monochrome display. I have also successfully run BetterBASIC on a Leading Edge PC/XT, a PCjr, and a Hyperion PC with no difficulty.

GW BASIC Program Conversion

To put it bluntly, there is no easy conversion from GW BASIC to BetterBASIC. There is a simple procedure outlined at the end of the manual that involves editing an ASCII version of the original. The keywords SOURCE and ENDFILE are added at the start and end of the file for compatibility with BetterBASIC's LIST ALL ASCII storage format. In addition, variable

Z80 System Owners Join The 16/32 Bit Revolution Through Evolution

Starting At
\$695.00



CO-PROCESSING

The most cost effective way for Z80 system owners to obtain 16/32 bit processing power and software compatibility is via the HSC CO-16 Attached Resource Processor.

CO-16 is compatible with any Z80 system running CPM 2.2 or CPM 3.

A few examples include:

- KAYPRO 2/4/10 • TRS 2/3/12/16
- AMPRO LITTLE BOARD
- HEATH 89 • SUPERBRAIN
- XEROX 820 • TELEVIDEO 802/803
- MORROW • EPSON QX-10
- LOBO • OSBORNE 1/EXEC
- CROMEMCO • Plus many more

CO-16

Every CO-16 is delivered with

- 16/32 bit micro processor • 16 bit Operating System • 256 Kilo RAM
- Z80 interface • 16 bit RAM disk driver • CPM80 2.2 RAM disk driver
- CPM 2.2 or CPM 3 compatibility
- sources with tools • hardware diagrams • board level or case with power supply.

CO-1686

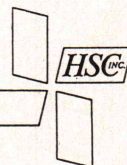
The only Z80 16 bit co-processor includes • INTEL 8086 • 6Mhz no wait states • MSDOS 2.11 • IBM BIOS emulator • Memory expansion to 768K • 8087 math co-processor • 3-channel Real Time Clock • Runs many IBM PC applications • Shares hard disk space with CPM80 • PC diskette compatibility on many systems • CPM86 • Concurrent CPM is coming.

CO-1668

The only Z80 16/32 bit co-processor includes • MOTOROLA 68000 microprocessor • 6 Mhz no wait states • CPM68K • Full "C" compiler with UNIX V7 library and floats • Memory expansion to 1.25 million bytes • NS16081 math co-processor • Real Time Clock • Complete software development environment • 100% file compatible with CPM80 • OS9/68 UNIX look alike coming in February.

Dealer, Distributor and OEM's invited

Hallock Systems Company, Inc.
267 North Main Street
Herkimer, N.Y. 13350
(315) 866-7125



Circle no. 26 on reader service card.

declarations must be made at the start of the file. If you don't declare the variables, they will be defined on use, but all integers will be defined as reals with the resultant loss of speed and precision that comes from the use of the real data type.

I was able to convert with no difficulty trivial programs such as the *Byte* benchmarks, and one fairly lengthy musical program came over without much modification. However, large programs such as PC-TALK or RBBS-PC would be easier to re-write from scratch with BetterBASIC than they would be to convert. This is because of major philosophical differences in the use of procedures and the use of structures for file I/O rather than *FIELDs* in an I/O buffer. Conceptually, it would probably be easier to translate a Pascal or C program than an existing GW BASIC program.

Refer to Table 3 for a complete summary of the differences between the two BASICs.

The Documentation

If there are two words that describe the BetterBASIC documentation, they are massive and thorough. The manual is an IBM-style 5½ by 8-inch slip-cased binder with over 550 pages! It is broken up into seven sections: General Information, Introduction to BetterBASIC, Quick Reference, Syntax Visuals, Appendices, and two new additions for v1.1—Chain/Overlays and Syscalls.

The syntax visuals are comparable to the IBM BASIC manual with one

or more pages in alphabetical order to describe each command and show examples of proper usage. The introductory chapter is very well done with an excellent tutorial. It is geared to beginners, but will provide a refresher for experienced BASIC programmers. It includes in the lesson on recursion a discussion of static versus dynamic storage of variables that will be useful to many microcomputer programmers. The presentation of variable scope and side effects in the procedure section will be helpful to those programmers whose only previous experience is with other BASICs. Appendices include an ASCII code table, keyboard scan codes, interrupts, character sets, module usage, module defined statements, use of Assembler, error codes, and GW BASIC conversion hints.

User Support

Because Summit Software is still a small company, any phone calls in search of support will be handled by one of the program's developers. This kind of direct access can be wonderful, especially when problems involve the more complex aspects of the system. There is no charge for phone support for registered users and Summit can be reached during business hours.

Complaints

As in all new products, there is still room for improvement. My main gripe has to do with errors in the manual. Until I received my v1.1 upgrade, I could find no reference to the *PUBLIC* keyword that is necessary for pre-

paring to *MAKE MODULEs*. There also seems to be some confusion when conceptual words from other languages are used interchangeably with BetterBASIC keywords; structure and record are used in this fashion, although record is never defined to mean the same as structure.

Some of the error codes that the compiler can generate are similar to the warnings of other compilers. I'd like to see the compiler add some of the missing parentheses that it can detect, rather than just notifying the programmer. Also, you cannot specify a program on the command line to be executed by BetterBASIC. This precludes the use of *B SETCLOCK* to read your clock/calendar in an *AUTOEXEC.BAT* file.

There is no provision made for tracing program execution, as *TRON/TROFF* are not supported. I hope the people at Summit are listening and provide a trace that, unlike GW BASIC's primitive version, can also show variable contents, not just line numbers.

My final complaint has to do with price—there is a hidden cost that a potential developer must keep in mind. If you plan to market the software you develop with BetterBASIC, you will need to purchase the Runtime System for an additional \$250 so your customers will be able to run your program without the necessity of purchasing their own copy of BetterBASIC.

Conclusions

So who will find this package useful?

	BetterBASIC	GW BASIC
BENCH1.BAS—write 64K bytes out to a file	14.6	43.3
BENCH2.BAS—read 64K bytes in from a file	10.4	29.1
BENCH3.BAS—5000 floating point operations	92.9(BCD) 37.2(BNY)	69.4
BENCH4.BAS—calculate all primes from 1 to 7000	35.2	208.1

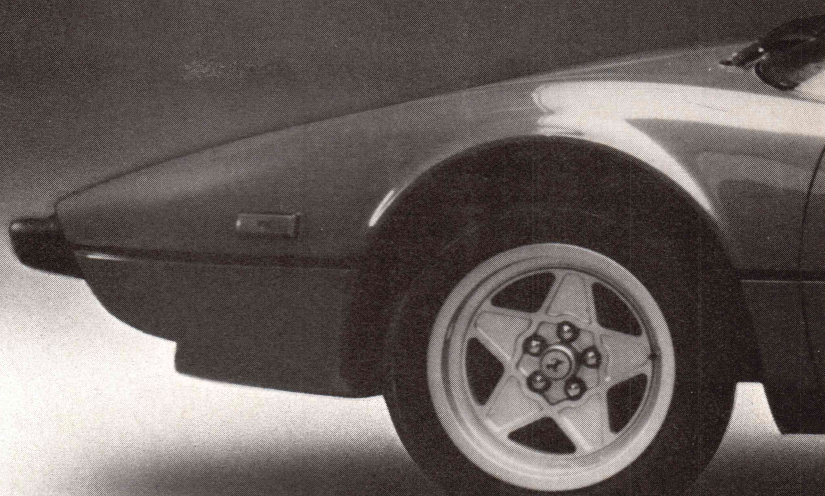
Table 4
Performance Benchmarks (All times in seconds)

Probably not the C wizards and Forth gurus of the world. They will feel restrained by the line numbering and the lack of mysterious, syntactic constructions. Pascal programmers will feel right at home with procedures that have local variables and statements like READLINE. I think this package will find its place mostly among programmers who are currently developing vertical market (i.e., business) software in BASIC.

A large percentage of software that is created for vertical markets, such as insurance agencies, medical practices, video rental clubs, and accounting firms, is written in some form of the BASIC language. One reason that is often given for this disproportionate popularity is that BASIC is available in some form on most personal computers, thus providing a greater market when the software is ported to a new machine. However, because of the widespread availability of Pascal and C compilers for micros in recent years, I think this reason is less important than the fact that much vertical market software is not developed by trained professional programmers. Often an accountant/car salesman/real estate broker/farmer who has learned a little BASIC on a personal computer realizes a need for some industry-specific piece of software. This combination of industry expertise and programming experience can result in a very successful program.

In the words of Ivar Wold, Summit's president, these people are faced with the "Pascal Dilemma"—they intend to learn Pascal or C but haven't found the time for it yet. With this package, a BASIC programmer will be able to write better, more maintainable code without the learning curve that is associated with learning a new language from scratch. After using BetterBASIC, it will be easy to pick up Pascal with very little effort, as all of Pascal's concepts can be found in BetterBASIC. However, BetterBASIC may provide just the excuse never to have to learn a new language again.

DDJ



Finally, A Lint and Make for MS™ – DOS

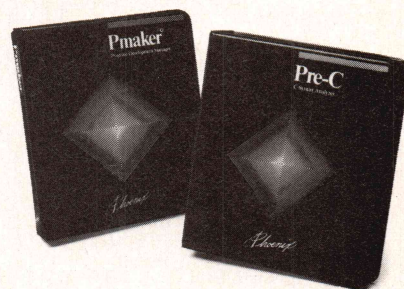
Get the full range of features C programmers working in UNIX™ have come to expect from their Lint and Make utilities. With Pre-C™ you can detect structural errors in C programs five times faster than you can with a debugger. Find usage errors almost impossible to detect with a compiler. Cross-check multiple source files and parameters passed to functions. Uncover interface bugs that are difficult to isolate. All in a single pass. Capabilities no C compiler, with or without program analyzing utilities, can offer. Pre-C outlints Lint, since you can handle analyses incrementally.

Pre-C's flexible library approach lets you maintain continuity across all programs in your shop, whether you use Pre-C's pre-built libraries, functions you already have, or some you might want to buy.

Plus, you're not limited to one particular library. Pre-C keeps track of all the libraries you're using to make sure that code correctly calls them.

With Pmaker™ you can update and track every module in your program. When you make a change in any source or include file, all you do

is run Pmaker. It will recompile changed modules and relink your program. With any compiler or linker you choose. Pmaker can update an object module library when one or several of the object modules are changed. You can use Pmaker to handle any task when a change requires several steps.



Pre-C by Phoenix. \$395. Pmaker by Phoenix. \$195.

Call (1) 800-344-7200.
In Massachusetts (617) 762-5030.

Or, write: Phoenix Computer Products, Corp., 1420 Providence Highway, Suite 115, Norwood, MA 02062.

Phoenix

PROGRAMMERS' PFANTASIES BY PHOENIX

Pre-C and Pmaker are trademarks of Phoenix Computer Products Corporation.
MS-DOS is a trademark of Microsoft Corporation. UNIX is a trademark of Bell Laboratories.

Sample assembly language interface to a BetterBASIC program

```
org      0

;--- define the Symbol Table

sym1     db      sym2-$      ;link to next symbol table entry
          db      'EXCHANGE' ;name of the Procedure
          dw      0          ;fill word ( MUST be here !!)
          dw      header1    ;pointer to Procedure header

sym2     db      sym3-$      ;link to next symbol table entry
          db      'FILL'     ;name of the Procedure
          dw      0          ;fill word ( MUST be here !!)
          dw      header2    ;pointer to Procedure header

sym3     db      0           ;Symbol Table Terminator

;-----

          even              ;headers must be on
                           ; WORD boundary

header1  dw      154h        ;MUST be 154h
          dw      exchange   ;pointer to actual asm procedure
          dw      2          ;number of Arguments
          dw      1          ;type of argument 0 (Integer)
          dw      1          ;type of argument 1 (Integer)

header2  dw      154h        ;MUST be 154h
          dw      fill       ;pointer to actual asm procedure
          dw      2          ;number of Arguments
          dw      3          ;type of argument 0 (String)
          dw      1          ;type of argument 1 (Integer)

;-----

; These are the procedures proper

exchange proc far           ;MUST be a far proc
.
.                           ; the EXCHANGE code
.
exchange endp

fill     proc far
.
.                           ; the FILL code
.
fill     endp
```

End Listing



**the
source debugger
for lattice C**

Your time and convenience come first! The MSD C Debugger™ is the last, and perhaps final, word in programming assistance for Lattice C users. C Debugger produces a high level view of C programs via function names, line numbers, variable names and C data types, plus a low-level view of machine addresses and instructions for testing assembler language functions.

More features include:

- All documentation is prepared for programmers.
- Online help screen throughout the process.
- Capability to single step through your program.
- Set break points, examine registers and variables.

\$165.00 + \$3.50 shipping

MSD

To order, call or write:
MICRO-SOFTWARE DEVELOPERS, INC.
214½ W. Main St. • St. Charles, IL 60174
312/377-5151

Lattice C is a trademark of Lattice, Inc.

Circle no. 134 on reader service card.



The Tools You Need To C You Thru.

Now the *WizardWare*™ Applications Programmers Toolkit provides everything you need to increase your C programming productivity.

APT™ features include:

- COMPLETE SOURCE CODE (over 5000 lines!)
- File handling with direct & keyed access
- Screen and Report Generators, with full screen handling for your programs
- Generic Terminal Driver for portable code
- String math functions, and string manipulation routines
- Reference Manual on Disk (over 50 pages)
- Tutorial Manual (over 25 pages) with Source for Mailing List Manager
- A host of useful Utilities, Database and File Editors
- Available for Lattice C, Mark Williams C, DeSmet C, BDS C, others.

Also Available: C-STARTER Toolkit, great for learning C!! Includes: Customized APT, DeSmet C Compiler, and "Programming in C on the IBM-PC" (200 pages)

APT/MS-DOS versions	\$495
APT/DeSmet C version	\$395
APT/BDS C version	\$395
C-Start (binary APT, DeSmet Compiler and Book)	\$295
APT/Manual only	\$ 50

Detailed Brochures on request

*Manual Cost will be applied if APT purchased within 30 days (\$10 re-stocking charge.) U.S. funds only, please.

Trademarks: MS-DOS: Microsoft; Lattice C: Lattice, Inc.; MWS: Mark Williams Co.; DeSmet C: C Ware, CI C&C Computer Innovations, Inc.; BDS C: BDS Software; DR. C: Digital Research; WizardWare, APT, C-Start: Shaw - American Technologies.

Call (502) 583-5527

Ask for APT™ or C-Start, or Send Check to:
Shaw ☆ American Technologies

WizardWare™

**830 South Second St. • Box 648
Louisville, KY 40201, USA**

(C.O.D. and Foreign Orders - Add \$5 Shipping/Handling)

References: Bank of Louisville, Citizens Fidelity Bank, Louisville Chamber of Commerce

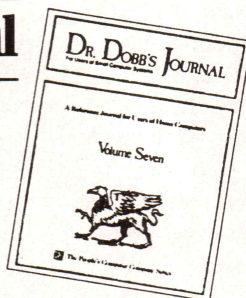
 

Circle no. 86 on reader service card.

Dr. Dobb's Journal

Bound Volumes

**Every Issue Available
For Your Personal Reference.**



We are pleased to offer this special discounted price to DDJ readers who order directly from us. From the nostalgia of Volume One—with authors like Steve Wozniak, Dennis Allison, Sol Libes, and more—to the technical maturity of Volume Seven, **Dr. Dobb's Journal Bound Volumes** are the ideal addition to your reference collection. They contain many issues which are no longer available and you get twelve issues for the price of seven individual back issues!

Send \$26.75 for volume 1, \$27.75 each for volumes 2-6, \$30.75 for volume 7, or \$165 for all seven and SAVE!

Please add the following per book: \$2.50 for UPS, \$1.25 for U.S. Mail, or \$3.25 for Foreign surface mail. Foreign Airmail rates available on request. Delivery times are one week for UPS or 6-10 weeks for U.S. or Foreign Mail.

Mail to: DDJ, 2464 Embarcadero Way, Palo Alto, CA 94303

Please allow 6-9 weeks for delivery.

Please provide a street address rather than a P.O. Box.

103

Microprocessor Software Development on VAX or PDP-11.



You can develop software for Z80, 8080, 8085, NCS800, and 8086 using native mode compilers and assemblers.

Use low-cost cross tools for other microprocessors. Interface in-circuit emulators perfectly. You can run Intel development tools under ISIS or UDI.

Our plug-in processor cards let you run CP/M-80, CP/M-86, or MS-DOS from any terminal on your VAX or PDP-11 system.

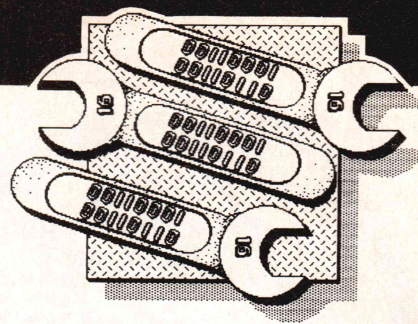
Prices start at just \$1295. Ask for our FREE catalog of 350 development and cross development tools.


Decmation

3375 Scott Blvd., Suite 236
Santa Clara, CA 95054
(408) 980-1678

Registered Trademarks: VAX, PDP: Digital Equipment Corporation; CP/M-80, 86: Digital Research; MS-DOS: Microsoft Corporation; ISIS; UDI: Intel.

Circle no. 19 on reader service card.



by Ray Duncan

68000 Square Root Routine

Jim Cathey of Spokane, Washington, writes: "To help [eliminate] the dearth of 68000 programming goodies you were mentioning, here is an integer square root program. It takes only about 10–11 times as long to execute as a divide instruction. I found this algorithm a few years back in *EDN* . . . I thought it was a nifty trick, and saved a clipping for when I needed it. Two muls instructions, one add.l, and this subroutine implement a true 16-bit vector length subroutine, in less than 2000 cycles (250 microsec. at 8 MHz)!"

"The original program (and my first attempt) had a bug, in that the intermediate variables were half the size of the input data. This works until the last trip through the loop, where you may have overflow errors that could ruin the accuracy. For speed, the main loop could be unrolled using word intermediates for the first 15 passes, and long on the last loop-equivalent. This should save some time if it's a problem. Of course, if speed is really a problem, one of the approximations for vector length you presented a while back would be many times faster." See Listing One (page 122) for the source code for Jim's 68000 square root routine.

MacFeedback

Jim Howell of San Jose, California, a regular correspondent to *DDJ*, writes: "I am writing regarding your recent rather disparaging comments (*DDJ*, December 1984, #98) on the Macintosh. I believe it is much too early for you to be forecasting the demise of the Mac. So 'only' 200,000 Mac's were delivered in its first year. How

many IBM PC's were delivered during its first year? And have you forgotten that even a year after the IBM PC came out, people were still complaining about the lack of software for it? As for comparing the Mac's 'lackluster sales' with the Lisa, I would guess that there are already more Macs than Lisas.

"One point I will agree on is the difficulty of developing software for the Mac. If Apple really wants lots of software for it, they need to make the required technical documentation available (and at a reasonable price). Development tools (compilers and assemblers) need to be able to run using a single Mac. There was an ad in *Infoworld* recently for a Mac assembler 'that requires no other hardware or software.' I don't know much about the availability of other language processors for the Mac, though I suspect (and hope) that most of them will be usable with only a single Mac. As for 'weak, bug-ridden' high-level languages, again consider the IBM PC. That's probably a fair description of many of the language processors that were available for the IBM PC during its first year. IBM's Pascal compiler, for example, is atrocious.

"I also agree that a 68000 assembler should be able to run with 128K and one disk drive, but comparing that to the DRI 8080 assembler that ran in 32K is very unfair. 8080 assembly language is much simpler than that of the 68000. The 68000 has more addressing modes, more instruction formats, and its assembly language has a more complex syntax. Also, the DRI 8080 assembler must have been a bare-bones, absolute assembler. A bare-bones 68000 assembler could be written to run on a 64K CP/M system (but probably not on a 32K system). A similar assembler

should run, therefore, on a 128K Mac. However, I would assume that Apple does not want to put out just a bare-bones assembler. If, for example, the Apple assembler generates object code that can be linked with separately compiled or assembled modules, this would add considerable complexity to the assembler.

"While a Mac assembler should be able to run in 128K, I don't think it's a big problem if it does not. Anyone doing serious software development on the Mac will have to have 512K (the next step after 128K) and two disk drives, even if some of the tools run in 128K. After all, how many developers of IBM PC software have only 128K in their machines? And even fewer have only one disk drive.

"I will be like the other readers who requested 68000 material and not submit any 68000 code. Like most of these readers (probably), I am interested in the 68000, but do not actually have a 68000 system. Therefore, I also do not have any 68000 code to submit. With the attitudes expressed in your column about the 68000 (and about Intel's 80286), one wonders whether you would want to include any 68000 code or material anyway.

"Finally [in reference to another column on identifying IBM PC environments], location F000:FFFEH on my Corona contains 0FEH, just like the PC/XT. Apparently, they want software to think it's running on an XT, just in case a hard disk happened to be attached (yes, I do own an IBM compatible, though I like my 6809 system better). The AT&T computer that we use at work has 00 (zero) at that location (the date in the ROM is 05/03/84)."

This letter from Jim neatly illustrates the polarization that the Mac has induced within the programming

community. There is a certain class of people who are dazzled by Apple's "insanely great" publicity and will forgive the Macintosh anything. There is another group, among whom I guess I must number myself, who find Apple's pretentious advertising and grandiose claims a little ridiculous, considering that almost everything that's good about the Mac's user interface was "borrowed" directly from the Xerox Star that has been around for ten years. Speaking of pretentious, see the recent *Playboy* interview with Steve Jobs for some amusing pronouncements, especially about us programmers over 30 years old.

The fact is that everybody would find the sales of 200,000 Macs in the first year to be very remarkable, if Apple hadn't spouted off all those projections about how it was going to build a Mac every 15 seconds in their new super MacFactory. To argue that the Mac has already sold more machines than the Lisa is not what you'd call a devastating rejoinder, since Apple has already thrown in the towel on the Lisa and reincarnated it as the Macintosh XL.

As for the Macintosh assembler, I feel that I'm on very solid ground with my previous comments, having written several assemblers myself. The fact that the instruction set of the 68000 is powerful and extensive cuts both ways: of course it means that you have to worry about assembling more complex syntax, but it also means that you have a much more powerful language in which to *write* the assembler. Unless, of course, you are so bull-headed as to insist on writing your assembler in a high-level language like Pascal, which is completely unsuited to the job. Then you *do* end up with monstrosities like the Apple Macintosh Assembler or the Microsoft 8086 Macro Assembler. The proof of my contention has been conveniently provided by the product "MacASM" from Mainstay of Agoura Hills, California. This is an integrated full screen editor, macro assembler, and resource compiler for the Macintosh that runs nicely in 128K and costs only \$125.00. "MacASM" is not copy protected and supports multiple

drives including hard disks. Mainstay can be reached at (818) 991-6540.

MSDOS Device Drivers

Stan Mitchell of San Jose, California, writes: "I thought I would pass along a tool that comes in handy when working with MSDOS device drivers. It depends upon a useful property of the NUL device driver. Unlike other devices, it is embedded in the DOS module (IBMDOS.COM for the PC) and its device header is at the head of the device chain. By using

the next device pointer in successive headers, the complete chain can be revealed. It is interesting that if any installed devices are present, the NUL header points to the first of these. When installed devices are exhausted, the link continues with the resident devices (contained in IBM-BIO.COM for the PC).

"The only 'trick' in the program is using the OPEN function to return the device header pointer. This is an undocumented feature of MSDOS."

Readers, please note that the format of the reserved area in the file

CP/M-80 C Programmers . . .

Save time

. . . with the BDS C Compiler. Compile, link and execute *faster* than you ever thought possible!

If you're a C language programmer whose patience is wearing thin, who wants to spend your valuable time *programming* instead of twiddling your thumbs waiting for slow compilers, who just wants to work *fast*, then it's

time you programmed with the BDS C Compiler.

BDS C is designed for CP/M-80 and provides users with quick, clean software development with emphasis on systems programming.

BDS C features include:

- Ultra-fast compilation, linkage and execution that produce directly executable 8080/286 CP/M command files.
- A comprehensive debugger that traces program execution and interactively displays both local and external variables by name and proper type.
- Dynamic overlays that allow for run-time segmentation of programs too large to fit into memory.
- A 120-function library written in both C and assembly language with full source code.
- Plus . . .
- A thorough, easy-to-read, 181-page user's manual complete with tutorials, hints, error messages and an easy-to-use index — it's the perfect manual for the beginner and the seasoned professional.
- An attractive selection of sample programs, including MODEM-compatible telecommunications, CP/M system utilities, games and more.
- A nationwide BDS C User's Group (\$10 membership fee — application included with package) that offers a newsletter, BDS C updates and access to public domain C utilities.

Reviewers everywhere have praised BDS C for its elegant operation and optimal use of CP/M resources. Above all, BDS C has been hailed for its remarkable *speed*.

BYTE Magazine placed BDS C ahead of all other 8080/286 C compilers tested for fastest object-code execution with all available speed-up options in use. In addition, BDS C's speed of compilation was almost *twice* as

fast as its closet competitor (benchmark for this test was the Sieve of Eratosthenes).

"I recommend both the language and the implementation by BDS very highly."

Tim Pugh, Jr.

in *Infoworld*

"Performance: Excellent. Documentation: Excellent. Ease of Use: Excellent."

InfoWorld

Software Report Card

"... a superior buy ..."

Van Court Hare
in *Lifelines/The Software Magazine*

Don't waste another minute on a slow language processor. Order your BDS C Compiler today!

Complete Package (two 8" SS&D disks, 181-page manual): **\$150**
Free shipping on prepaid orders inside USA.
VISA/MC, COD's, rush orders accepted.
Call for information on other disk formats.

BDS C is designed for use with CP/M-80 operating systems, version 2.2 or higher. It is not currently available for CP/M-86 or MS-DOS.

BDSoftware

BD Software, Inc.
P.O. Box 2368
Cambridge, MA 02238
(617) 576-3828

Circle no. 12 on reader service card.

Engineered for Excellence



We put more into the Davong DataSystem™

so you can get more from your Personal Computer.*

You are interested in more productivity. Davong designed the DataSystem for convenience, versatility, and reliability.

Convenience is built-in. Like the package design that slips under your monitor. And installation that's a snap. Put one adapter in a short slot. Connect one cable. Plug it in and go. And you don't need to be an expert to use our software. Menu-driven programs with simple commands and on-line information put you at ease.

Versatility is unsurpassed. DataSystem tape accepts true DOS commands (COPY, etc.) so you can make file-by-file backups. You can even run programs from tape. Or, back up entire disk volumes in minutes with our Tape Manager program. Your data speeds to tape at one million characters a minute. Just data. DataSystem doesn't waste your tape space with non-data areas. And tape capacity is always guaranteed.

Choose from 10, 21, 32 or 43-megabyte disk sizes plus 24 megabytes of removable tape storage. Emulate an XT with Davong Fixed Disk software or enhance your computer's performance with optional Davong Multi-OS® software. And Multi-OS readies the DataSystem for Davong MultiLink™ networking.

Reliability is assured. Only fully-tested components are used in the DataSystem. Before it leaves quality control, each drive must pass days of rigorous system-level tests—and the rack. Each DataSystem must survive a torturous burn-in with no problems. Only then do we consider it good enough for your desk.

If you have the Data, Davong has the System for you. Convenience, versatility and reliability. Why settle for less?

See us at Comdex/Spring, Booth #3232.

*IBM® PC, XT, AT, and many compatibles

IBM® is a registered trademark of International Business Machines Corporation.
Multi-OS®, Davong DataSystem™, and Davong MultiLink™ are trademarks of Davong Systems, Inc.



Davong Systems, Inc.
217 Humboldt Court
Sunnyvale, CA 94089
Phone: (408) 734-4900
Telex: 176386

control block is different under MSDOS 2.0 than it is under version 3.0. Stan's C program and the assembly language source for the `_peek` function are provided as Listing Two (page 122) and Listing Three (page 125).

Microsoft Assembler Bug of the Month

Gregor Owen of Port Jefferson Station, New York, writes: "I don't know if anyone's still counting, but here's my contribution for Microsoft Assembler bug of the month. In the assembly below, the code `es:lods` produces an error; however, the code below it, `mov ax,es:[si+1]`, produces an error too. If you comment out the `es:lods`, both errors disappear—so superior to the old-fashioned kind of assembler, where frequently the removal of defective code only removes errors in the vicinity of the defect.

"At this point, I don't really know if `es:lods` represents a legal operation. I don't happen to have an Intel book at this location, and since every 8086/88 instruction has a unique set of addressing modes, it's difficult to get any kind of feel for what's allowed. Needless to say, the IBM/Microsoft assembler manual is not helpful. It's true that if one comments out `es:lods` and assembles `lods byte ptr es:[si+0]` both errors disappear, but I don't think anybody who's used the Microsoft assembler for any short while would take any comfort from that.

"And while we're on the subject, notice the expressive elegance and beauty of the phrase

```
lods byte ptr es:[si + 0]
```

typical of the language Intel has provided for us. The `si+0`, I concede, is something I've been doing ever since I managed to assemble some kind of complicated expression that had `[si]` in it—producing code that referenced `[si+0FFFFH]` and didn't generate an assembler error. Of course I probably don't have to use `si+0` in every expression, but who wants to spend the time to find out? That's the beauty of the language: every programmer, in order to produce stuff that works at

all, will develop a private dialect containing much myth and legend, thus making the source even more incomprehensible than it is anyway.

"My personal theory about 8086/88 assembly language is that what we have here is a marketing coup on the part of Intel. They had a ridiculous processor, with the traditional inept Intel approach to registers and addressing. Somehow, they sensed it would be awfully embarrassing to expose this thing in all the stark simplicity of their 'lxi h,7'-type assembler [for the 8080 among other processors]. So they commissioned the finest, most convoluted, Pascalized minds in the software industry to come up with an assembler so grand and structured and incredibly intricate that no one would ever notice how awful the thing was for which it was assembling. Judging by the results, they seem to have succeeded.

"And those of us who labor in the vineyards of the information revolution can't be all too high-horsey about this: after all, the Intel/Microsoft language makes it almost impossible for those pesky amateurs to figure anything out, thus preserving the world of 8086/88 assembler for us, the elect: people paid to spend the necessary endless hours figuring out how to use the language and avoid the Microsoft bugs. A sort of 'programmer's employment project' for the new age."

Gregor's program is provided as Listing Four (page 126). My uneducated guess, in this case, is that the Microsoft Assembler interprets the `es:lods` portion of the statement `es:lods` as the definition of a label named `es`. It then gives you the error message "symbol already different type," since the name `es` is already hardwired into the symbol table as an assembler directive. All the other errors follow from the ensuing type conflicts.

Fun with your PC/AT

Assemble the following instruction sequence into a short assembly language program and execute it on the IBM PC/AT of your choice:

```
mov bx,0ffffh
```

Pascal and C Programmers

Your programs can now compile the **FirstTime™**

FirstTime is an intelligent editor that knows the rules of the language being programmed. It checks your statements as you enter them, and if it spots a mistake, it identifies it. *FirstTime* then positions the cursor over the error so you can correct it easily. *FirstTime* will identify all syntax errors, undefined variables, and even statements with mismatched variable types. In fact, any program developed with the *FirstTime* editor will compile on the first try.

More than a syntax checker!

FirstTime has many unique features found in no other editor. These powerful capabilities include a zoom command that allows you to examine the structure of your program, automatic program formatting, and block transforms.

If you wish, you can work even faster by automatically generating program structures with a single key-stroke. This feature is especially useful to those learning a new language, or to those who often switch between different languages.

Other Features: Full screen editing, horizontal scrolling, function key menus, help screens, inserts, deletes, appends, searches, and global replacing.

Programmers enjoy using *FirstTime*. It allows them to concentrate on program logic without having to worry about coding details. Debugging is reduced dramatically, and deadlines are more easily met.

FirstTime for PASCAL	\$245
FirstTime for C	\$295
Microsoft PASCAL Compiler	\$245
Microsoft C Compiler	\$395
Demonstration disk	\$25

Get an extra **\$100 off** the compiler when it is purchased with **FirstTime**. (N.J. residents please add 6% sales tax.)

Spruce
Technology Corporation
110 Whispering Pines Drive
Lincroft, N.J. 07738
(201) 741-8188 or (201) 663-0063

Dealer enquiries welcome. Custom versions for computer manufacturers and language developers are available.

FirstTime is a trademark of Spruce Technology Corporation.



Circle no. 65 on reader service card.

mov ax,[bx]

The system dies immediately. The same sequence of instructions will run just fine on an 8086 or 8088-based machine. What is happening here?

It turns out that on the 80286, accessing a word operand at offset 0FFFFH, i.e., a segment wrap, causes

a hardware interrupt 13H. Alas (as Jerry Pournelle would say), this interrupt is already used by the IBM PC ROM BIOS for the disk driver. So when your program accidentally generates a segment wrap fault, it traps to the disk driver with nonsense parameters in the registers and probably writes garbage all over your nice

hard disk. This kind of phenomenon, when we move a previously happy program from an IBM PC to a PC/AT and watch it go to heaven, helps us while away those rainy afternoons.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 196.

16-Bit Toolbox (Text begins on page 118)

Listing One

```
* Integer Square Root (32 to 16 Bit)
*
* (Exact method, not approximate)
*
* Call with:
*     D0.L = Unsigned number
*
* Returns:
*     D0.L = SQRT (D0.L)
*     D1.L <> 0 if not exact root
*
* Uses:
*     D1-D4 as temporaries ---
*     D1 = Error term
*     D2 = Running estimate
*     D3 = High bracket
*     D4 = Loop counter
*
* Notes:
*     Result first in D0.W, but is valid in longword.
*     Takes from 1480 to 1832 cycles (including RTS).
*     (Word version is from 548 to 660 cycles).
lsqrt    move.w    #15,d4    ; Loop count (bits-1 of result)
          moveq    #0,d1    ; Result in D1
          moveq    #0,d2
sqrt1    asl.l     #1,d0    ; Get 2 leading bits at a time and
          roxl.l   #1,d1    ; into Error term for extrapolation.
          asl.l     #1,d0    ; (Classical method, easy in binary)
          roxl.l   #1,d1
          asl.l     #1,d2    ; Running estimate * 2
          move.l    d2,d3
          asl.l     #1,d3
          cmp.l     d3,d1
          bls      sqrt2    ; New error term > 2* running est.?
          addq.l    #1,d2    ; Yes, we want 1 bit then.
          addq.l    #1,d3    ; Fix up new error term.
          sub.l     d3,d1
sqrt2    dbra      d4,sqrt1  ; Do all 16 bit-pairs.
          move.l    d2,d0    ; Returns answer in D0.W
          rts
```

End Listing One

Listing Two

"C" source code for Stan Mitchell's program to dump the device driver chain.

```
1  /*****
2  * dd_dump.c          program for displaying device driver chain *
3  *                   for DOS 2.00,2.10,3.00                       *
4  *                   *                                           *
5  * by stan mitchell   november 21,1984                           *
6  *                   Lattice C v. 1.04                           *
7  *****/
8
9  #define void int
10 #define byte char
11 #include "stdio.h"
12
```

(Continued on page 124)

DSD 80

FULL SCREEN SYMBOLIC DEBUGGER

**"THE SINGLE BEST DEBUGGER
FOR CP/M-80. A TRULY
AMAZING PRODUCT."**

LEOR ZOLMAN
AUTHOR OF BDS C

- ☐ Complete upward compatibility with DDT
- ☐ Simultaneous instruction, register, stack & memory displays
- ☐ Software In-Circuit-Emulator provides write protected memory, execute only code and stack protection.
- ☐ Full Z80 support with Intel or Zilog Mnemonics
- ☐ Thirty day money back guarantee
- ☐ On-line help & 50 page user manual

**NOW
ONLY \$125.**

SOFTADVANCES

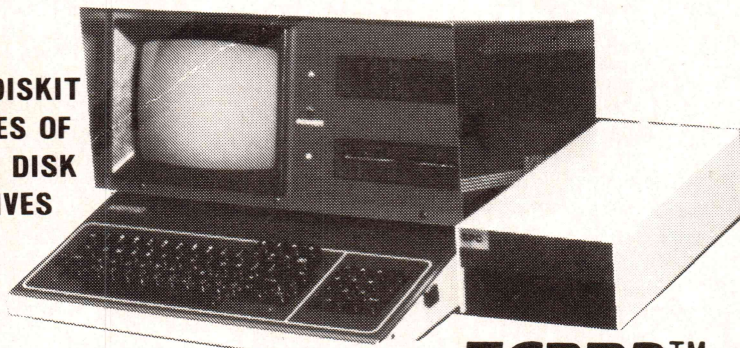
P.O. BOX 49473 AUSTIN, TEXAS 78765 (512) 478-4763



Circle no. 63 on reader service card.

FOR THE SERIOUS KAYPRO® USER

THE DISKIT
SERIES OF
HARD DISK
DRIVES



...now with **ZCPR3™**

Now you can add from 5 to 40 Megabytes of fast-access Winchester storage to your KAYPRO 2, 4, or 10. The DISKIT is only 4 inches high; 5.7 if you get the two drive model with the *removable* 5 or 10 Mb. cartridge, and weighs less than 10 pounds. Easily disconnect DISKIT from the computer whenever you want, and if more capacity is required, just swap your drive for a larger model.

Our DISKIT Model 10 has 10.8 Megabytes of *formatted* capacity . . . 20% more than a Kaypro 10, and runs about twice as fast. Installs in minutes. Call SPC now and ask for more information. Quantity and prepayment discounts are available.

SYSTEMS PERIPHERALS CONSULTANTS

9747 Business Park Avenue
San Diego, CA 92131
(619) 693-8611

Circle no. 104 on reader service card.

RUN/C:™ The Affordable C Interpreter

Available NOW for only \$149.95!

Finally, a painless introduction to the C language. With **RUN/C: The C Interpreter** you can create and run C language programs in an environment as easy to use as BASIC.

RUN/C is C for the rest of us. It is a robust implementation of standard K&R. **RUN/C** is for both the beginner and professional.

RUN/C includes full floating point, 8087 support, structures, unions, casts and more than 100 built-in C functions.

With **RUN/C** you get all this with a command structure modeled after BASIC's using familiar terms such as EDIT, RUN, LIST, LOAD, SAVE, TRON, SYSTEM, etc.

Since **RUN/C** is a true interpreter it means that C programs can be written, tested and run within a single protected environment. It is a teaching tool and a source code debugger.

Here's more good news . . .

- Great documentation: a 400-page, easy-to-read manual filled with executable programs
- Array-index and pointer bounds checking
- Variable-trace and dump diagnostics PLUS an integral program profiler
- Full buffered and unbuffered file I/O
- Printer and asynch support
- Forking to your favorite full screen editor with automatic return to **RUN/C** with your edited program
- System Requirements: IBM® PC or compatible with PC-DOS 2.0 or MS™-DOS 2.0 or greater with ANSI.SYS.

Get things right the first time with **RUN/C: The C Interpreter.™**

For immediate delivery or more information, call:

1-800-847-7078

(in N.Y. 1-212-860-0300)

or write: Lifeboat Associates™
1651 Third Avenue
New York, NY 10128

Circle no. 74 on reader service card.

TURBO GRAPHICS™

for use with

Borlands Turbo Pascal™

A comprehensive set of Pascal procedures and binary routines that includes most of the graphics capabilities found in BASICA in addition to several new graphics routines.

MAJOR FEATURES

(on medium resolution graphics screen)

- Draw large letters, circles, and lines in color.
- Create complicated graphics using DRAW "macros".
- Create and use WINDOWS non-destructively.
- Paint enclosed areas. Get and Put screen memory.
- Scroll any portion of the screen up or down.
- Use the font editor to create letters and graphics characters.
- Combine features to create interesting animations.

Catalog Numbers:

UTIL-100-IBM PC \$39.95
UTIL-100-IBM PCjr each

Shipping and Handling:

\$5.00 each



Diversified Educational Enterprises, Inc.
725 MAIN STREET
LAFAYETTE, IN 47901
317-742-2690

Circle no. 118 on reader service card.

16-Bit Toolbox (Listing Continued, text begins on page 118)

Listing Two

```

13  /* DOS function calls */
14  #define OPENFCB  0x0f
15  #define CLOSEFCB 0x10
16  #define VERSION  0x30
17
18  extern int _peek();
19
20  struct DEVHDR
21  {
22      unsigned nxthdr_off; /* doubleword ptr to next device hdr in chain */
23      unsigned nxthdr_seg;
24      unsigned attr;      /* type of device driver */
25      unsigned strat;     /* device strategy entry point */
26      unsigned intrpt;    /* interrupt entry point */
27      char dname[8];      /* device name */
28  };
29
30  struct FCB
31  {
32      byte drive;          /* drive designator */
33      char fname[11];      /* file or device name */
34      unsigned curblk;     /* current blk (set to 0 by OPENFCB) */
35      unsigned recsiz;     /* logical record size (set to 0x80 by OPENFCB) */
36      long fsize;          /* file size in bytes */
37      unsigned date;       /* creation or last update date */
38      byte sys_rsv[10];    /* fields reserved for DOS */
39      byte bset[5];        /* relative record numbers */
40  };
41
42  struct RSV2_X
43  {
44      unsigned time;       /* creation or last update time */
45      byte attribute;      /* device or file attribute */
46      unsigned dhdr_off;   /* offset address of device header */
47      unsigned dhdr_seg;   /* segment address of device header */
48      byte unk[3];         /* unknown usage */
49  };
50
51  struct RSV3_X
52  {
53      unsigned time;       /* creation or last update time */
54      unsigned attribute;   /* device or file attribute */
55      unsigned dhdr_off;   /* offset address of device header */
56      unsigned dhdr_seg;   /* segment address of device header */
57      byte unk[2];         /* unknown usage */
58  };
59
60  struct DEVHDR device;    /* device header */
61  struct FCB device_fcb;   /* standard FCB for device */
62  struct RSV2_X *dos2x;    /* reserved field definitions DOS 2.x */
63  struct RSV3_X *dos3x;    /* reserved field definitions DOS 3.x */
64
65  main()
66  {
67      int dev_off, dev_seg, i, j;
68
69      initfcb(0, "NUL", ""); /* set up standard FCB for NUL device */
70      if (bdos(OPENFCB, &device_fcb) & 0xff) /* open the device */
71      {
72          printf("Unable to open device\n");
73          exit(1);
74      }
75      if ((bdos(VERSION, 0) & 0xff) == 3)
76      { /* the reserved fields are allocated differently in DOS 3.x */
77          dos3x = (struct RSV3_X *) &device_fcb.sys_rsv[0];
78          dev_off = dos3x->dhdr_off;
79          dev_seg = dos3x->dhdr_seg;
80      }
81      else if ((bdos(VERSION, 0) & 0xff) == 2)
82      { /* ... than they were in DOS 2.x */
83          dos2x = (struct RSV2_X *) &device_fcb.sys_rsv[0];
84          dev_off = dos2x->dhdr_off;
85          dev_seg = dos2x->dhdr_seg;

```

```

86     }
87     else
88     { /* forget it for DOS 1.10 or earlier */
89         printf(" DOS 2.00 or newer required\n");
90         exit(1);
91     }
92
93     printf(" Device driver chain .... \n");
94     printf("\n");
95
96     /* display the current DOS chain of device drivers */
97     printf(" ptr      type      name      strategy ptr  interrupt ptr  \n");
98
99     for (;;)
100     {
101         printf("%04x:%04x ",dev_seg,dev_off); /* device header ptr */
102         if (dev_off == 0xffff) break; /* if last in chain, offset=0xffff */
103         /* move the device header into the data segment structure "device" */
104         _peek(dev_seg,dev_off,&device,sizeof(device));
105         printf(" %04x ",device.attr); /* display the device attribute word */
106         if ((device.attr & 0x8000) == 0) /* if a block device ... */
107             for (j=0;j<3;j++) printf("%02x",device.dname[j]); /* there is no device name */
108         else
109             for (j=0;j<7;j++) printf("%c",device.dname[j]); /* otherwise display device name */
110
111         printf(" %04x:%04x ",dev_seg,device.strat); /* strategy entry point */
112         printf("%04x:%04x \n",dev_seg,device.intrpt); /* "interrupt" entry point */
113         dev_seg=device.nxthdr_seg; /* set up ptr to next device header */
114         dev_off=device.nxthdr_off; /* and loop back to display it */
115     }
116
117 }
118
119 /* initialize standard FCB */
120 void initfcb(drv,name)
121 byte drv;
122 char name[];
123 {
124     int i;
125
126     device_fcb.drive=drv; /* drive designation: default drive=0, A=1, etc. */
127     for (i=0;i<10;i++) device_fcb.fname[i]=name[i]; /* device or file name */
128     for (i=0;i<4;i++) device_fcb.bset[i]=0; /* fields not zeroed by open call */
129 }
130

```

End Listing Two

Listing Three

Assembly language source for "peek" library function,
used in Stan Mitchell's device driver dump program.

```

PAGE
PAGE 62,132
NAME peek
;*****
;
; void _peek(segment,offset,buffer,nbytes)
;
; unsigned segment; /* segment portion of memory addr */
; unsigned offset; /* offset portion of memory addr */
; byte *buffer; /* local memory buffer (in data segment) */
; unsigned nbytes; /* number of bytes to transfer */
;
; Lattice C assembly language interface convention followed
; (ES=DS on entry)
;*****
;
= 0004 EXTRA EQU 4
;
PGROUP GROUP PROG
PROG SEGMENT BYTE PUBLIC 'PROG'
PUBLIC _PEEK
ASSUME CS:PGROUP
;

```

(Continued on next page)

16-Bit Toolbox (Listing Continued, text begins on page 118)

Listing Three

```

0000          _PEEK PROC NEAR
0000 55          PUSH BP          ;
0001 8B EC      MOV BP,SP        ;
0003 1E          PUSH DS        ;
0004 56          PUSH SI        ;
0005 57          PUSH DI        ;
0006 51          PUSH CX        ;
          ;
          ;get source segment
0007 8E 5E 04   MOV DS,WORD PTR [BP+EXTRA]
          ;
          ;get source offset
000A 8B 76 06   MOV SI,WORD PTR [BP+EXTRA+2]
          ;
          ;get destination offset in ES
000D 8B 7E 08   MOV DI,WORD PTR [BP+EXTRA+4]
          ;
          ;get byte count for transfer
0010 8B 4E 0A   MOV CX,WORD PTR [BP+EXTRA+6]
          ;
          ;move the bytes into local buffer
0013 F3/ A4     REP MOVSB
          ;
0015 59          POP CX          ;
0016 5F          POP DI          ;
0017 5E          POP SI          ;
0018 1F          POP DS          ;
0019 5D          POP BP          ;
001A C3          RET            ;
001B          _PEEK ENDP
PROG          ENDS
END

```

End Listing Three

Listing Four

Gregor Owen's Microsoft Assembler Bug of the Month.
 Assemble this brief program for some entertaining error messages,
 then comment out the line "es:lodsb" and assemble it again.

```

microwiff segment
          assume cs:microwiff,ds:microwiff,es:microwiff,ss:microwiff
microwimp:
          es:lodsb
          lods      byte ptr es:[si+0]
          mov       ax,es:[si+1]
microwiff ends
          end       microwimp

```

End Listings

NEW! Advanced Trace86™

Symbolic Debugger & Assembler Combo

- Full-screen trace with single stepping; Even backstepping!
- Write & Edit COM & EXE programs
- Conditional breakpoints (programmable)
- Switch between trace and output screen; Or set up two monitors
- 8087, 80186, 80286, 80287 support
- Write labels & comments on code
- Polish hex/decimal calculator
- and more... Priced at \$175.00

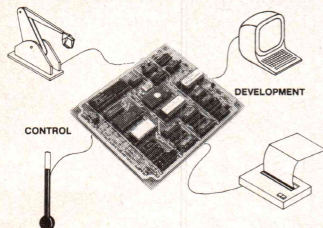
To order or request more information contact:

M Morgan Computing Co., Inc.

P.O. Box 112730, Dallas, TX 75011
(214) 739-5895

Circle no. 128 on reader service card.

THE IPC-SBC88 DEVELOPMENT AND CONTROL SYSTEM



WRITE IT — RUN IT — ROM IT

A single board computer development and control system that is so simple to use, you will be developing applications programs the first day!

- Choice of Basic or FORTH in ROM
- Onboard EPROM programmer for complete program development
- 8 channel, 8 bit analog to digital converter
- RS-232 terminal and parallel printer port for program entry
- Two 8 bit input ports
- Two 8 bit output ports
- Two 8 bit sinking outputs rated at 500 mA, 50 VDC
- Time of day
- Up to 32 K of user memory
- 8088 16 bit uP
- Assembled and tested S279 Kits start at \$79

MasterCard and Visa accepted

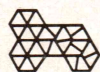
Vesta Technology, Inc. 7100 W. 44th Ave. Suite 101
Wheat Ridge, CO 80033 (303) 422-8088

Circle no. 124 on reader service card.

BYSO™ LISP

has features that will delight both beginners and advanced programmers. A fast, reliable and complete interpreter for the IBM PC and true compatibles. \$125 includes 100 pg. ref. manual and application notes that put you months ahead on useful projects (making a hybrid language with C, accessing system functions and I/O ports, building your own dialect, etc.).

LEVIEV INSTRUMENT CO.



P.O. Box 31B
McDowell, VA 24458
703-396-3345

IBM PC is a trademark of the IBM Corp.

Circle no. 25 on reader service card.

PERFORMANCE ACCELERATORS FOR CP/M-80 & MP/M-80

WSOPTION for WORDSTAR 3.0 & 3.3 (installs itself right into WordStar)

FAST

Speeds up action of WordStar functions so you spend less time waiting for WordStar to take action on your commands.

PRODUCTIVE

A superior print-while-edit capability is included. You can now use one terminal or micro to edit one file while printing another without reducing your typing speed!

CONVENIENT

At print time you can select 1 of 4 printers under CP/M or 1 of 8 printers under MP/M. Under MP/M you do a proper attach and detach of the printer so it is always free when you are not using it. You are informed if the printer is in use at print time so you do not hang up. Many additional features installed via menu.

MPPLUS for MP/M-80

A group of programs and modules designed to improve your console and disk response under MP/M-80. A performance increase of 2 to 3 times is usual for disk I/O.

Send \$35.00 for each item plus \$2.00 postage.

INCLUDE DISK FORMAT REQUIRED

Continuum Microsystems Ltd. Use your Visa
21 McCarty Crescent or M.C.
Markham, Ontario
Canada L3P 4R4 (416) 294-8536
WordStar reg. MicroPro, CP/M MP/M reg. D.R.I.

Circle no. 29 on reader service card.

ICs PROMPT DELIVERY!!! SAME DAY SHIPPING (USUALLY)

OUTSIDE OKLAHOMA: NO SALES TAX

DYNAMIC RAM			
256K	256Kx1	150 ns	\$ 5.77
64K	64Kx1	120 ns	2.30
64K	64Kx1	150 ns	1.69
64K	64Kx1	200 ns	1.87
EPROM			
27C256	32Kx8	250 ns	\$21.25
27256	32Kx8	250 ns	\$18.75
27128	16Kx8	250 ns	9.37
27C64	8Kx8	200 ns	8.75
2764	8Kx8	250 ns	3.97
2732A	4Kx8	250 ns	4.69
2716	2Kx8	450 ns	3.21
STATIC RAM			
6264LP-15	8Kx8	150 ns	\$10.50
6116LP-3	2Kx8	150 ns	2.67

OPEN 6 1/2 DAYS: WE CAN SHIP VIA FED-EX ON SAT.

MasterCard/VISA or UPS CASH COD

Factory New, Prime Parts

MICROPROCESSORS UNLIMITED

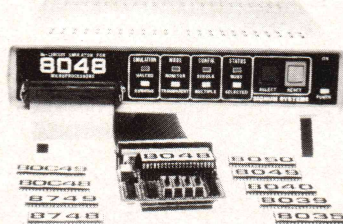
24,000 S. Peoria Ave., BEGGS, OK. 74421 (918) 267-4961

Prices shown above are for March 26, 1985

Please call for current prices. Prices subject to change. Please expect higher or lower prices on some parts due to supply & demand and our changing costs. Shipping & insurance extra. Cash discount prices shown. Orders received by 6 PM CST can usually be delivered to you by the next morning, via Federal Express Standard Air @ \$6.00; Priority 1 @ \$11.50

Circle no. 64 on reader service card.

IN-CIRCUIT EMULATOR For 8050, 8049 and 8048 µC's



For Hardware/Software Design the E232-48 replaces the target microcomputer for complete emulation under control of the host computer. It emulates all of the above µC's plus their ROM-less and CMOS versions. It's features are:

- ★ Real time emulation up to 11 Mhz.
 - ★ Full 4K Emulation Memory.
 - ★ Hardware Breakpoints.
 - ★ In-line Assembler and Disassembler.
 - ★ Upload, Download & Terminal mode drivers for IBM-PC, CP/M-80 and CP/M-86 are included.
- Cross Assemblers for 8048 series and other µP's running on the above host systems-From \$150
E232-48 in-circuit emulator..... \$1795

SIGNUM SYSTEMS

726 Santa Monica Blvd
Santa Monica, CA 90401 (213) 451-5382

Circle no. 106 on reader service card.

No source code for your REL files?

REL/MAC

converts a REL file in the Microsoft™ M80 format to a ZILOG™ or 8080 source code MAC file with insertion of all public and external symbols.

- REL/MOD lists library modules
- REL/VUE displays the bit stream
- 50 page manual with examples
- free brochure available
- REL/PAK includes all of the above

REL/PAK for 8080 only \$99.95

REL/PAK for Z80 & 8080 \$134.95

on 8"SSSD disk for CP/M™ 2.2

Send check, VISA, MC or C.O.D. to



**MICROSMITH
COMPUTER TECHNOLOGY**

P.O. BOX 1473 ELKHART, IN 46515

1-800-622-4070

(Illinois only 1-800-942-7317)

Circle no. 41 on reader service card.

Users' Group

Over 40 volumes of public domain software including:

- compilers
- editors
- text formatters
- communications packages
- many UNIX-like tools

Write or call for more details

The C Users' Group

415 E. Euclid • Box 97
McPherson, KS 67460
(316) 241-1065

Circle no. 17 on reader service card.

Now With Windowing! \$49.95 Basic Compiler

MTBASIC

Features:

- | | |
|--------------------|------------------|
| Multitasking | Windowing |
| Handles interrupts | Interactive |
| Fast native code | Compiles quickly |
| Floating point | No runtime fee |

MTBASIC is a true native code compiler. It runs Bytes's Sept. '81 seive in 26 seconds; interpreters take over 1400 seconds! Because MTBASIC is multitasking, it can run up to 10 Basic routines at the same time, while displaying ten separate windows. Pop-up/down menus are a snap to implement.

MTBASIC combines the best of interpreters and compilers. To the programmer, MTBASIC appears to be an extremely fast interpreter. MTBASIC compiles a typical 100 line Basic program in 1 second, yet it generates blindingly fast code. No more waiting for long compiles.

AVAILABLE for CP/M (Z-80) and PC-DOS systems.

ORDERING: Specify format when ordering. We accept Visa, MC, checks and COD. Send \$49.95 plus \$3.50 shipping and handling (\$10 overseas) to:



P.O. Box 2412 Columbia, MD 21045-1412
301/792-8096

Circle no. 88 on reader service card.

LISP FOR THE IBM PERSONAL COMPUTER.

THE PREMIER LANGUAGE
OF ARTIFICIAL
INTELLIGENCE FOR
YOUR IBM PC.

■ DATA TYPES

Lists and Symbols
Unlimited Precision Integers
Floating Point Numbers
Character Strings
Multidimensional Arrays
Files
Machine Language Code

■ MEMORY MANAGEMENT

Full Memory Space Supported
Dynamic Allocation
Compacting Garbage Collector

■ FUNCTION TYPES

EXPR/FEXPR/MACRO
Machine Language Primitives
Over 190 Primitive Functions

■ IO SUPPORT

Multiple Display Windows
Cursor Control
All Function Keys Supported
Read and Splice Macros
Disk Files

■ POWERFUL ERROR RECOVERY

■ 8087 SUPPORT

■ COLOR GRAPHICS


■ LISP LIBRARY

Structured Programming Macros
Editor and Formatter
Package Support
Debugging Functions
.OBJ File Loader

■ RUNS UNDER PC-DOS 1.1 or 2.0

IQLISP

5 1/4" Diskette
and Manual _____ \$175.00
Manual Only _____ \$ 30.00

 **Integral Quality**

P.O. Box 31970
Seattle, Washington 98103-0070
(206) 527-2918

Washington State residents add sales tax.
VISA and MASTERCARD accepted.
Shipping included for prepaid orders.

ADVERTISER INDEX

Reader Service No.	Advertiser	Page No.	Reader Service No.	Advertiser	Page No.
5	AFIPS	83	57	Micro Compatibles	71
1	Alpha Computer Service	82	32	Micro Cornucopia	3
16	Arity Corporation	43	30	Micro Direct International	99
7	Artisoft	69	134	Micro Software Developers	117
4	Ashton Tate	34-35	132	MicroMotion	109
130	Automata Design Associates	79	41	MicroSmith	127
10	Avocet Systems, Inc.	47	64	Microprocessors Unlimited	127
12	BD Software, Inc.	119	66	Mitek	105
11	Borland International	1	128	Morgan Computing	127
14	Borland International	C-4	54	Mullen Computer Products	69
8	Business Utility Software	71	76	New Generation Systems	93
17	C User's Group	127	59	Ordinate Solutions	73
18	C Ware	67	122	Phoenix Computer Products	10-11
21	Chalcedony	79	70	Phoenix Computer Products	115
9	CompuView	17	69	Plu Perfect Systems	65
22	Computer Helper Industries, Inc.	69	71	Poor Person Software	91
29	Continuum Microsystems Ltd.	127	73	Procode International	105
27	Creative Solutions	51	72	Programmer's Shop	21
28	D&W Digital	18	51	QCAD Systems	31
*	DDJ Classified Advertising	97	20	Quelo	88
135	Data Base Decisions	12	83	Raima Corp.	30
6	Datalight	65	79	Rational Systems, Inc.	77
2	Davong	120	80	Revasco	109
19	Decimation	117	78	SLR Systems	67
52	Digital Pathways	57	110	STSC	C-2
33	Digital Research Computers	63	85	SemiDisk Systems	39
118	Diversified Education Enterprises	124	86	Shaw American Technologies	117
35	Ecosoft, Inc.	81	106	Signum Systems	127
*	Edward Réam	82	63	Soft Advances	123
114	Emerald Systems	2	88	Softaid, Inc.	127
36	Essential Software	85	89	Softfocus	91
37	Faircom	77	90	Software Horizons, Inc.	81
40	Fox Software, Inc.	103	92	Softway, Inc.	87
13	General Communications Corp.	75	75	Solution Systems	37
*	Gimpel Software	26	93	Solution Systems	37
*	Gimpel Software	48	94	Solution Systems	37
43	Greenleaf Software, Inc.	23	95	Solution Systems	37
26	Hallock Systems Consultants	113	102	Solution Systems	59
44	Harvard Softworks	107	68	Solutionware	89
23	Hippopotamus Software	112	97	Speedware	91
46	IQ Software	15	65	Spruce Technologies Corp.	121
48	Illyes Systems	104	60	Starlight Forth Systems	65
91	Info Pro Systems	67	98	Summit Software	29
*	Integral Quality	128	104	Systems Peripheral Consultants	123
15	Integrand Research Corp.	73	82	Tatum Labs	109
50	Interface Technologies	27	56	Thunder Software	89
*	J. D. Owens & Assoc.	75	77	UniPress Software	7
55	Laboratory Microsystems Inc.	81	87	Unlimited Processing	49
58	Lattice, Inc.	75	124	Vesta Technology, Inc.	127
53	Level 5 Research	105	112	Wendin, Inc.	9
25	Levien Inst. Company	127	116	Wizard Systems	91
74	Lifeboat Associates	123	126	Zeducorp	24
24	M'Guinness Design	73	*	DDJ Back Issues	79
*	MIX Software	C-3	*	DDJ Bound Volume	117
62	Manx Software	25	*	DDJ C Compiler	77
84	Megamax, Inc.	71			

Advertising Sales Offices

East Coast
Walter Andrzejewski (415) 424-0600

Midwest/West Central
Michele Beaty (317) 875-0557

Northern California/Northwest
Shawn Horst (415) 424-0600

Southern California/Southwest
Beth Dudas (714) 643-9439

Classified Advertising
Alice Abrams (415) 424-0600

Advertising Director
Stephen Friedman (415) 424-0600

Advertising Coordinator
Lisa Boudreau (415) 424-0600

Assistant Advertising Coordinator
Jay Horvath (415) 424-0600

Introducing the MIX Editor

(with Split Screen - both horizontal and vertical)

A Powerful Addition To Any Programmer's Tool Box

Full Screen Editing
WordStar Key Layout
Custom Key Layouts
Terminal Configuration
Help Files
Backup Files

Introductory Offer
Only

29⁹⁵

30 Day Money Back Guarantee

Programmable
Macro Commands
Custom Setup Files
Mnemonic Command Mode
Multiple File Editing
Split Screen Editing

For PC DOS/MSDOS (2.0 and above/128K) • IBM PC/Compatibles, PC Jr., Tandy 1000/1200/2000, & others
*For CPM80 2.2/3.0 (Z80 required/64K) • 8" SSSD, Kaypro 2/4, Osborne I SD/DD, Apple II, & others

Great For All Languages

A general purpose text processor, the MIX Editor is packed with features that make it useful with any language. It has auto indent for structured languages like Pascal or C. It has automatic line numbering for BASIC (255 character lines). It even has fill and justify for English.

Terminal Configuration

A utility for defining terminal features (smart features included) allows the editor to work with any terminal. Over 30 of the most popular terminals are built-in.

Custom Key Layouts

Commands are mapped to keys just like WordStar. If you don't like the WordStar layout, simply change it. Any key can be mapped to any command. You can also define a key to generate a string of characters, great for entering keywords.

Split Screen

You can split the screen horizontally or vertically and edit two files simultaneously.

Macro Commands

The MIX Editor allows a sequence of commands to be executed with a single keystroke. You can define a complete editing operation and perform it at the touch of a key.

MIX
software

2116 E. Arapaho
Suite 363
Richardson, Tx 75081
(214) 783-6001

MSDOS is a trademark of Microsoft
PCDOS is a trademark of IBM
CPM80 is a trademark of Digital Research
WordStar is a trademark of MicroPro

Custom Setup Files

Custom keyboard layouts and macro commands can be saved in setup files. You can create a different setup file for each language you use. The editor automatically configures itself using a setup file.

Command Mode

Command mode allows any editor command to be executed by name. It is much easier to remember a command name versus a complicated key sequence. Command mode makes it easy to master the full capability of the editor. Frequently used commands can be mapped to keys. Infrequent commands can be executed by name.

Editor Commands

The editor contains more than 100 commands. With so many commands, you might think it would be difficult to use. Not so, it is actually extremely simple to use. With command mode, the power is there if you need it, but it doesn't get in your way if you don't. Following is a list of some of the commands.

Cursor Commands

Left/Right/Up/Down
Tab Right/Tab Left
Forward Word/Backward Word
Beginning of Line/End of Line
Scroll Up/Scroll Down
Window Up/Window Down
Scroll Left/Scroll Right
Top of File/Bottom of File
• • •

Block Commands

Copy/Move/Delete
Read/Write
Lower Case/Upper Case
Fill/Justify
Print

File Commands

Directory (with wild cards)
Show File/Help File
Input/Output File
Delete File/Save File

Other Commands

Split Screen/Other Window
Find String/Replace String
Replace Global/Query Replace
Delete Line/Undelete Line
Delete Word/Undelete Word
Insert Mode/Overwrite Mode
Open Line/Join Line
Duplicate Line/Center Line
Set Tab/Clear Tab
• • •

To Order: Call Toll Free 1-800-622-4070, (Illinois only 1-800-942-7317)

Mix Editor ____ \$29.95 + shipping (\$5 USA/\$10 Foreign) Texas residents add 6% sales tax

Visa ____ MasterCard ____ Card # ____ Exp. Date ____

COD ____ Check ____ Money Order ____ Disk Format ____

Computer ____ Operating System: MSDOS ____ PC DOS ____ CPM80 ____

Name ____

Street ____

City/State/Zip ____

Country ____

Phone ____

MIX
software

2116 E. Arapaho
Suite 363
Richardson, Tx 75081

Dealer Inquiries Welcome
Call (214) 783-6001

They said it couldn't be done. Borland Did It. Turbo Pascal 3.0

**MOST SIGNIFICANT PRODUCT
OF THE YEAR - PC WEEK**

The industry standard

With more than 250,000 users worldwide Turbo Pascal is the industry's de facto standard. Turbo Pascal is praised by more engineers, hobbyists, students and professional programmers than any other development environment in the history of microcomputing. And yet, Turbo Pascal is simple and fun to use!

COMPILATION SPEED	8.1
EXECUTION SPEED	9 ^{SEC}
CODE SIZE	12 K
BUILT-IN INTERACTIVE EDITOR	YES
ONE STEP COMPILE (NO LINKING NECESSARY)	YES
COMPILER SIZE	39K
TURTLE GRAPHICS	YES
BCD OPTION	YES
PRICE	\$69 ⁹⁵

TURBO 3.0 TURBO 2.0 MS PASCAL

16	206
13 ^{SEC}	20 ^{SEC}
12 K	35 K
YES	NO
YES	NO
35K	300K+
NO	NO
NO	YES
\$54 ⁹⁵	\$295 ⁰⁰

The best just got better: Introducing Turbo Pascal 3.0

We just added a whole range of exciting new features to Turbo Pascal:

- First, the world's fastest Pascal compiler just got faster. Turbo Pascal 3.0 (16 bit version) compiles twice as fast as Turbo Pascal 2.0! No kidding.
- Then, we totally rewrote the file I/O system, and we also now support I/O redirection.
- For the IBM PC versions, we've even added "turtle graphics" and full tree directory support.
- For all 16 Bit versions, we now offer two additional options: 8087 math coprocessor support for intensive calculations and Binary Coded Decimals (BCD) for business applications.
- And much much more.

The Critics' Choice.

Jeff Duntemann, PC Magazine: "Language deal of the century . . . Turbo Pascal: It introduces a new programming environment and runs like magic."

Dave Garland, Popular Computing: "Most Pascal compilers barely fit on a disk, but Turbo Pascal packs an editor, compiler, linker, and run-time library into just 39K bytes of random-access memory."

Jerry Pournelle, BYTE: "What I think the computer industry is headed for: well documented, standard, plenty of good features, and a reasonable price."

Portability.

Turbo Pascal is available today for most computers running PC DOS, MS DOS, CPM 80 or CPM 86. A XENIX version of Turbo Pascal will soon be announced, and before the end of the year, Turbo Pascal will be running on most 68000 based microcomputers.

An Offer You Can't Refuse.

Until June 1st, 1985, you can get Turbo Pascal 3.0 for only \$69.95. Turbo Pascal 3.0, equipped with either the BCD or 8087 options, is available for an additional \$39.95 or Turbo Pascal 3.0 with both options for only \$124.95. As a matter of fact, if you own a 16-Bit computer and are serious about programming, you might as well get both options right away and save almost \$25.

Update policy.

As always, our first commitment is to our customers. You built Borland and we will always honor your support.

So, to make your upgrade to the exciting new version of Turbo Pascal 3.0 easy, we will accept your original Turbo Pascal disk (in a bend-proof container) for a trade-in credit of \$39.95 and your Turbo87 original disk for \$59.95. This trade-in credit may only be applied toward the purchase of Turbo Pascal 3.0 and its additional BCD and 8087 options (trade-in offer is only valid directly through Borland and until June 1st, 1985).

(*) Benchmark run on an IBM PC using MS Pascal version 3.2 and the DOS linker version 2.6. The 179 line program used is the "Gauss-Seidel" program out of Alan R. Miller's book: *Pascal programs for scientists and engineers* (Sybex, page 128) with a 3 dimensional non-singular matrix and a relaxation coefficient of 1.0.

TURBO PASCAL

NOT COPY-PROTECTED

Available at better dealers nationwide. Call (800) 556-2283 for the dealer nearest you. To order by Credit Card call (800) 255-8008, CA (800) 742-1133

Carefully Describe your Computer System!

Mine is: ☐ 8 bit ☐ 16 bit ☐ MS-DOS
 I Use: ☐ PC-DOS ☐ CP/M 86
 ☐ CP/M 80 ☐ CP/M 86
 My computer's name/model is: _____

The disk size I use is:

☐ 3 1/2" ☐ 5 1/4" ☐ 8"

Name: _____

Shipping Address: _____

City: _____ State: _____ Zip: _____

Telephone: _____

For update:
original Turbo
disk must
accompany
order

YES! I want the Best! Please send: _____ Quantity _____

Pascal 3.0 \$ 69.95 _____

Pascal w/8087 \$109.90 _____

Pascal w/BCD \$109.90 _____

Pascal w/8087 & BCD \$124.95 (SAVE \$24.90) _____

* These prices include shipping to all U.S. cities. All foreign orders add \$10 per product ordered.

Subtotal (CA 6% tax) _____

Trade-in Credit Claimed: _____

Amount Enclosed: _____

Payment: ☐ VISA ☐ MC ☐ BankDraft ☐ Check

Credit Card Expir. Date: _____

Card #: _____

COD's and Purchase Orders WILL NOT be accepted by Borland. California residents: add 6% sales tax.
 Outside USA: add \$10 and make payment by bank draft, payable in US dollars drawn on a US bank.

Circle no. 14 on reader service card.

BORLAND
INTERNATIONAL

Software's Newest Direction
4585 Scotts Valley Drive
Scotts Valley, CA 95066
TELEX 172373

Turbo Pascal is a registered trademark of Borland International, Inc.
PC Week is a trademark of Ziff-Davis Pub. Co.